DATS 6313\_10: Time series Analysis and Modelling

Dr. Reza Jafari

Final Term Project

Atharva Haldankar

George Washington University

05-03-2022

|  |  |  |
| --- | --- | --- |
| **INDEX** | | |
| **Chap.** | **Topic** | **Pg. No.** |
| 1 | Introduction | 5 |
| 2 | Description of the Data   1. pre-processing 2. Dependent variable vs Time 3. ACF/PACF plot 4. Correlation Matrix with Heat Map 5. Train-Test Split | 5-8 |
| 3 | Stationarity | 9-10 |
| 4 | Time series decomposition | 10-11 |
| 5 | Holt Winters Method | 11-12 |
| 6 | Feature Selection | 12-13 |
| 7 | Base Models   1. Average Method 2. Naïve method 3. Drift Method 4. SES method | 14-17 |
| 8 | Multiple Linear Regression (MLR) | 19 |
| 9 | ARMA, ARIMA & SARIMA model order selection. | 20 |
| 10 | Levenberg Marquardt Algorithm | 21-23 |
| 11 | Final Model Selection | 24 |
| 12 | h-step Prediction | 25 |
| 13 | Summary and conclusions | 26 |
| 14 | References | 26 |
| 15 | Appendix | 27-36 |

|  |  |  |
| --- | --- | --- |
| **No.** | **Figures** | **Pg. No.** |
| 1 | Plot for electricity usage every 60 minutes VS time | 7 |
| 2 | Change in the unit cost every month | 8 |
| 3 | ACF & PACF plot | 8 |
| 4 | Heat map for correlation amongst continuous variables. | 8 |
| 5 | ACF & PACF Plot for hourly electricity usage. | 9 |
| 6 | Rolling mean and rolling Variance | 9 |
| 7 | ADF and KPSS test for the data | 9 |
| 8 | STL decomposition | 10 |
| 9 | Plot for seasonality, trend and residual | 10 |
| 10 | Original Data vs Seasonally adjusted data. | 10 |
| 11 | Original data vs Detrended data | 11 |
| 12 | ACF for prediction error (Holt Winters) | 11 |
| 13 | Holt Winters method | 12 |
| 14 | Average method Forecasting | 14 |
| 15 | Naive method forecasting | 15 |
| 16 | Drift method forecasting | 16 |
| 17 | SES method forecasting | 17 |
| 18 | GPAC | 19 |
| 19 | ACF PACF for data after differencing | 19 |
| 20 | ACF for residuals (ARMA) | 20 |
| 21 | Train and prediction for ARMA(2,0) | 20 |
| 22 | ACF for ARIMA | 21 |
| 23 | ARIMA Train vs Prediction. | 22 |
| 24 | SARIMA train vs prediction | 23 |
| 25 | 25 h-step prediction for MLR | 24 |

**ABSTRACT**

This report focuses on developing a python code. The objective of this report is to apply the course learning objectives to a real dataset for modeling & prediction. A real world data is acquired and with the help of all the tools and knowledge from the learnt from this course is applied to find the best model to forecast.

**1: Introduction**

Time Series Analysis was performed on a dataset which records the electricity usage every 15 minutes from [Kaggle](https://www.kaggle.com/jaganadhg/house-hold-energy-data). The data was then transformed to hourly electricity usage. A time series decomposition was done on this data and plotted the seasonally adjusted data and the detrended data.

After the time series decomposition, different models are used to check the accuracy of each model and which model works the best for this data. The final model is then selected and the h-step prediction is then performed.

**2 : Description of the dataset**

Pre-processing of the data

The dataset is taken from Kaggle. First, the end time and the date column were combined and was set as the index.

After setting the index, a column named “season” was added, which indicates the season. This is a categorical column which consists of 4 categories (1,2,3,4) which indicates the 4 seasons of the year.

1: Spring

2: Summer

3: Fall

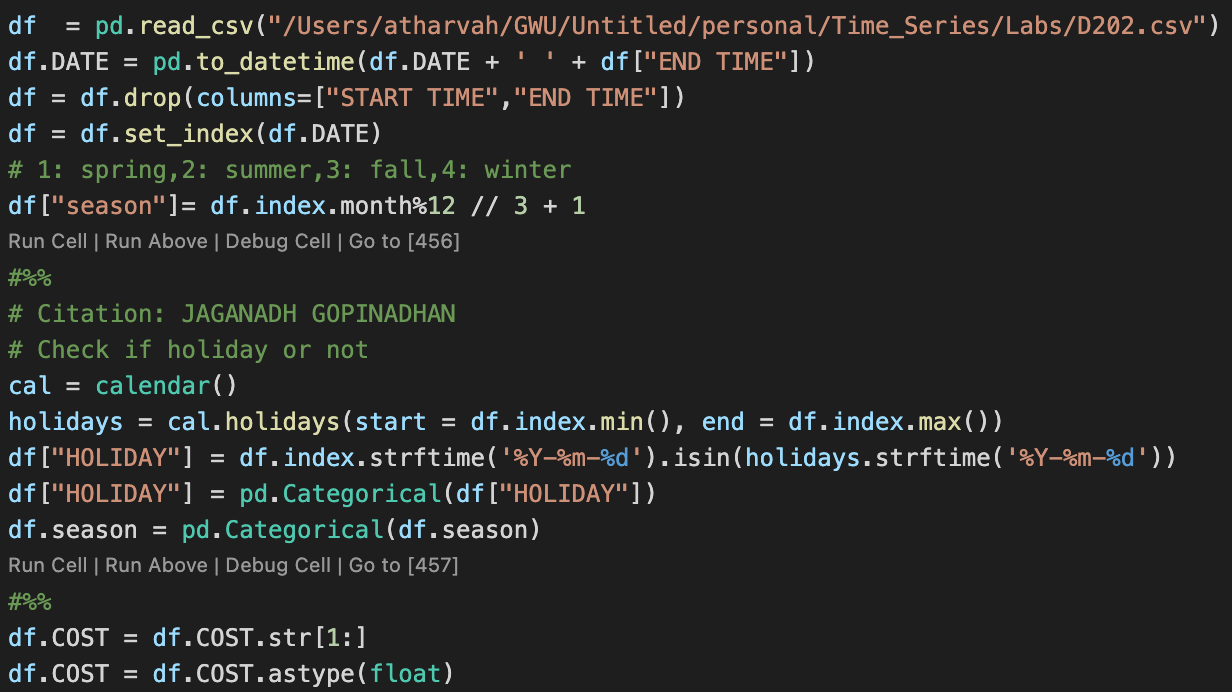
4: Winter

Another column named “HOLIDAY” was added, which indicates whether it was federal holiday or not. This column too is a categorical column which consists of two categories (0,1)

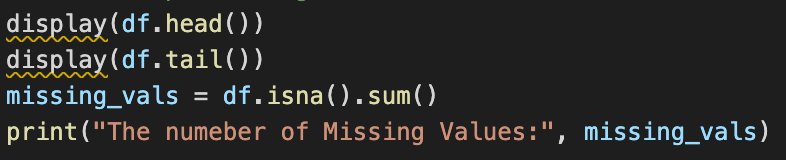
0: Not a Holiday

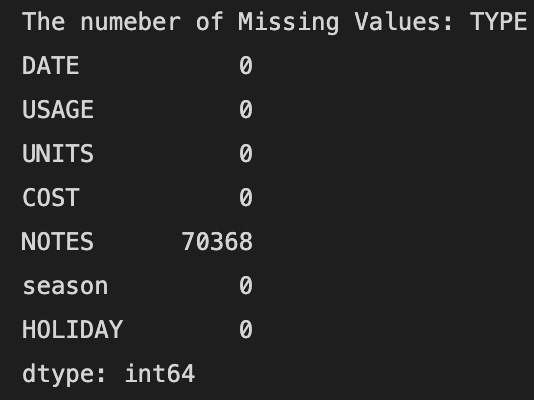
1: Holiday

The “COST” column records the cost of the electricity used. The cost was recorded as a string which was later changed to float.



Columns were then inspected for NA values. It was discovered that the “NOTES” columns only consisted of NA values while no other columns had any NA values.



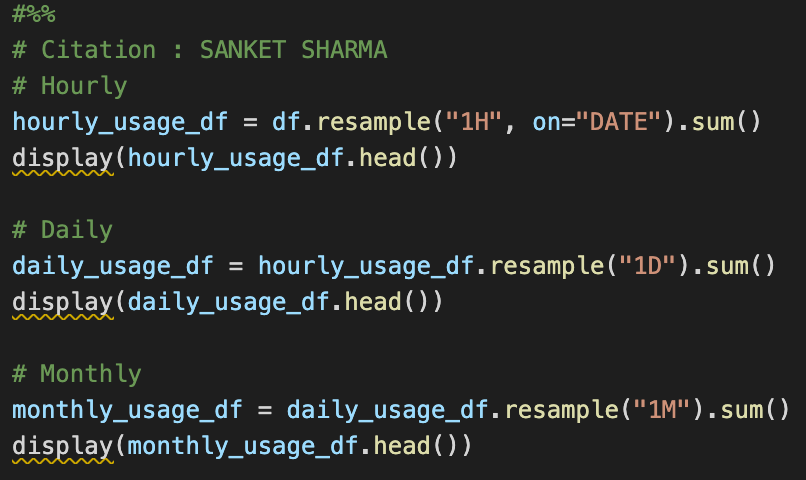


To have a better picture of the scenario, 3 new data frames were created which are derived from the original dataset.

hourly\_usage\_df: this records the electricity usage for every hour. This data frame is used for this project.

daily\_usage\_df: this records the electricity usage for 24 hours.

monthly\_usage\_df: this records the electricity usage monthly.



The same columns were added to these data frames along with “per\_unit\_cost” and “usage\_month”. These columns record the rate for electricity and the month of usage respectively.



Dependent variable and it’s plot with time

The dependent variable for this project is the electricity usage for every 1 hour.

The independent variables are season, holiday and cost.

The electricity usage for every 60 minutes is first plotted against time.

It can be clearly seen that this data data is additive.

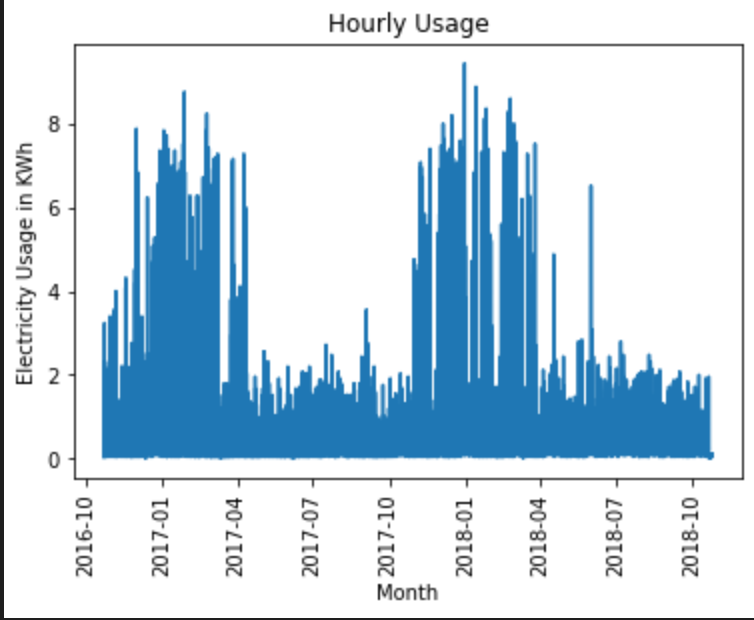


Figure 1: Plot for electricity usage every 60 minutes VS time

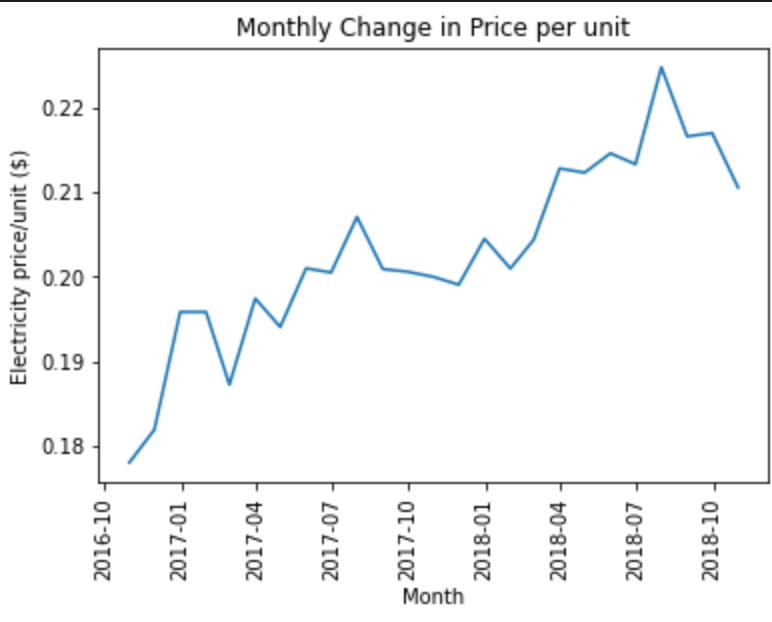
. 

Figure 2 Change in the unit cost every month

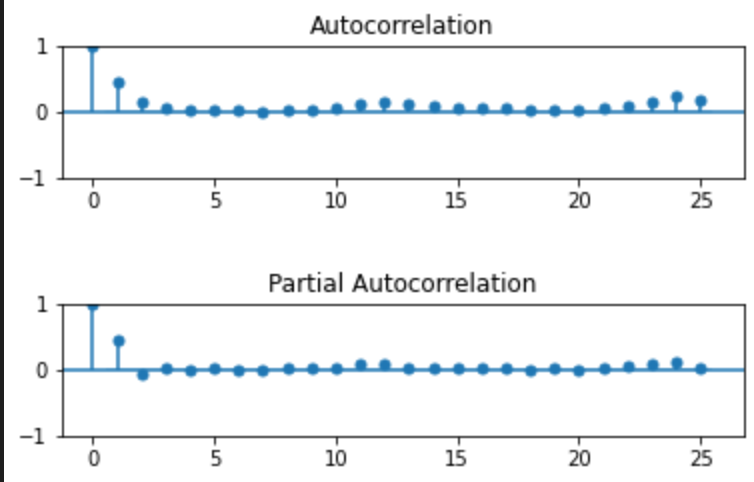


Figure 3 ACF & PACF plot

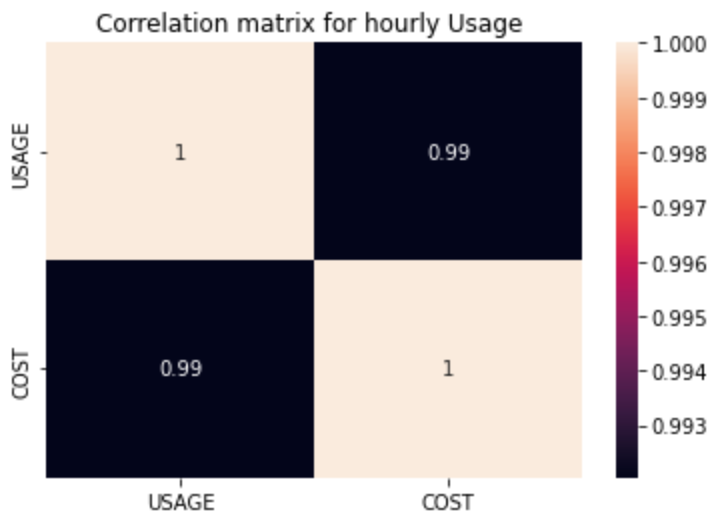


Figure 4 Heat map for correlation amongst continuous variables.

The hour\_usage\_df is then split for training and testing.

The training data consists of 80% and testing consists of 20%.



**3 : Stationarity**

From the figure below it can be seen that there is an immediate cutoff after lag 1 for both ACF & PACF plot indicating that the data is stationary.

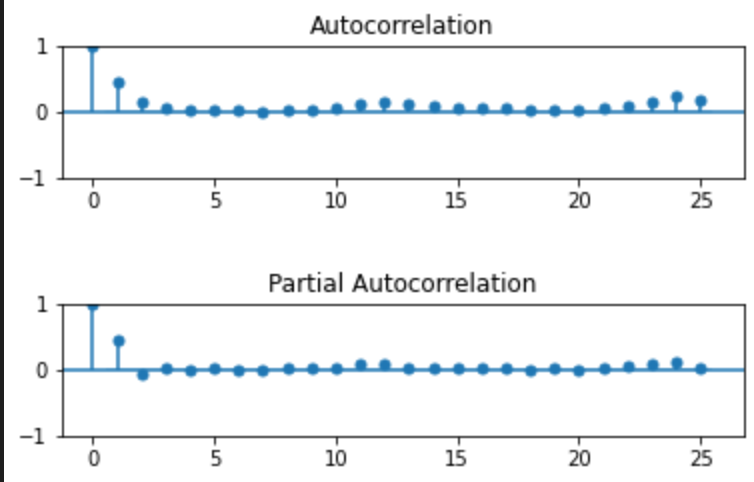
****

Figure 5 ACF & PACF Plot for hourly electricity usage.

The rolling mean and the rolling variance converges into a straight line which indicates that the data is stationary.

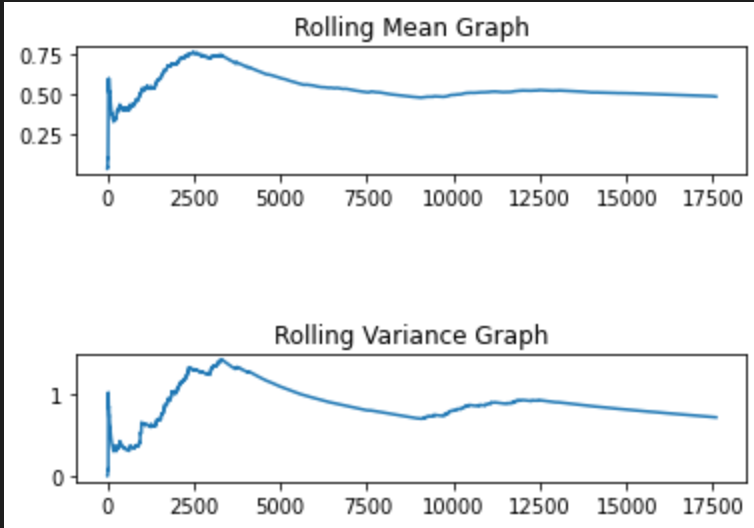


Figure 6 Rolling mean and rolling Variance

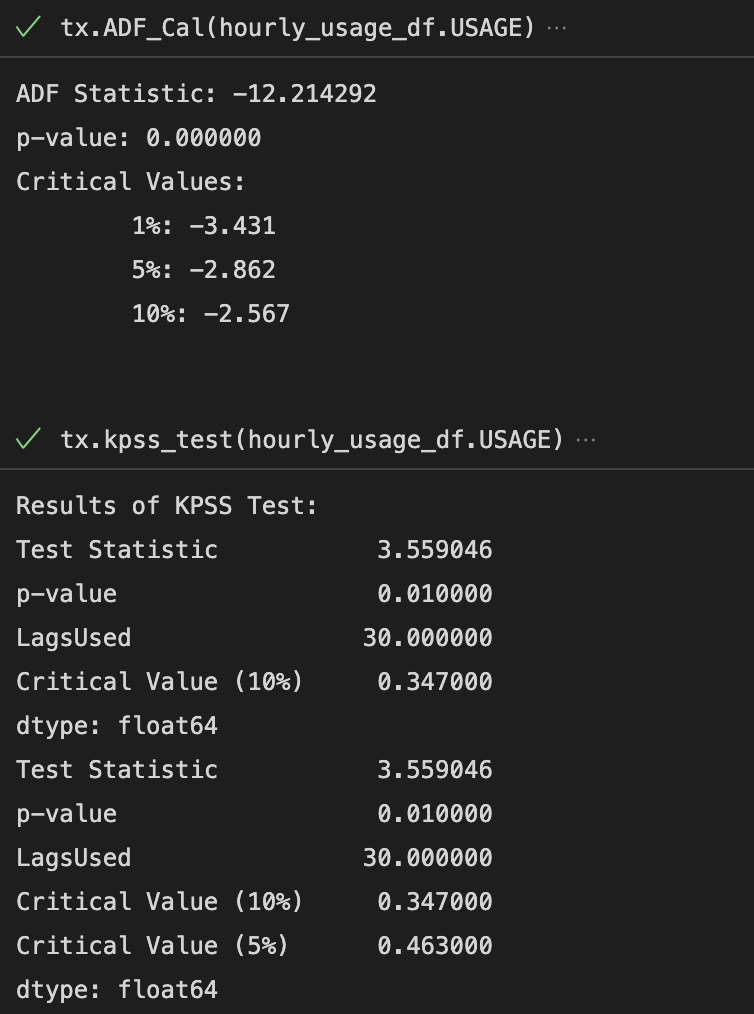


Figure 7 ADF and KPSS test for the data

The ADF and the KPSS test confirms that the data is stationary. The p value for ADF test is less than 0.05 which allows us to reject the null hypothesis and incorporate the alternate hypothesis which means the data is stationary. For the KPSS test the opposite is observed. Hence the null hypothesis cannot be rejected and the hence the data is stationary.

**4 : Time Series Decomposition**

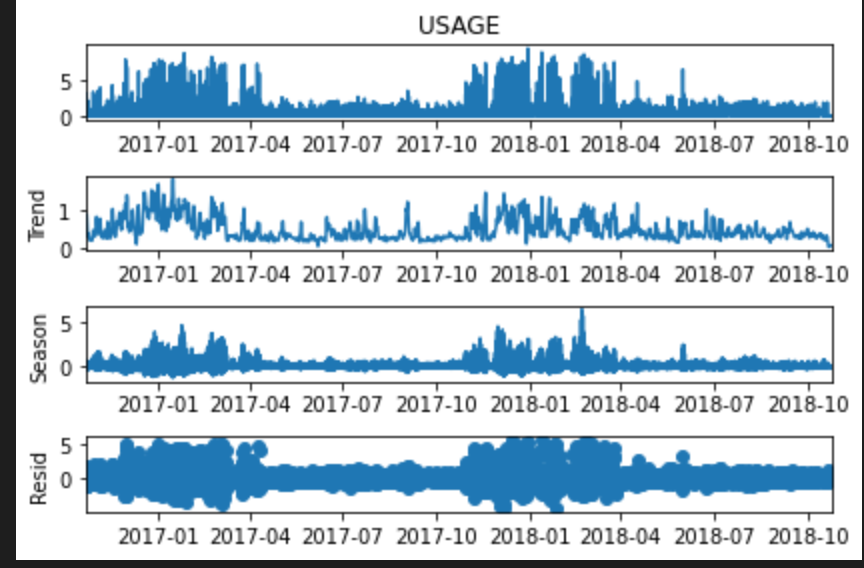


Figure 8 STL decomposition

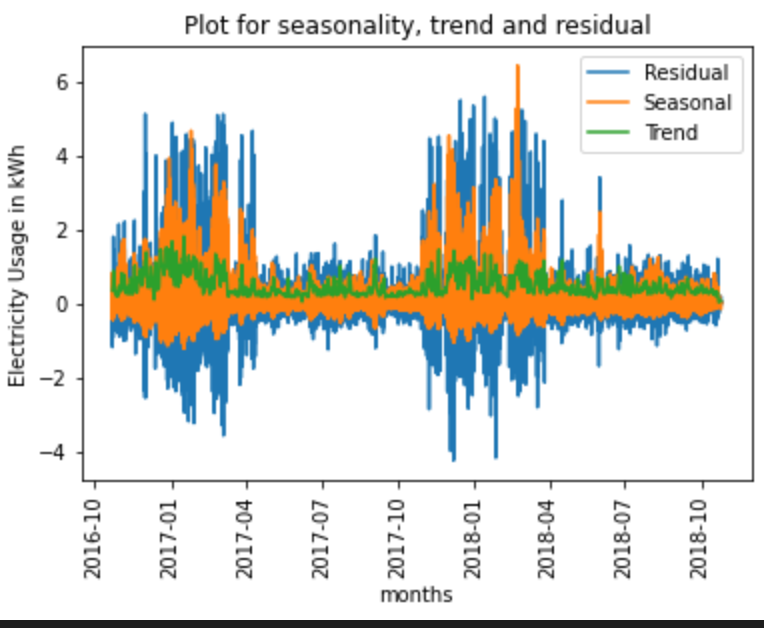


Figure 9 Plot for seasonality, trend and residual

Since the data is additive, for seasonally adjusted data, the seasonality is subtracted from the data and for detrended data, the trend is subtracted from the data as well.

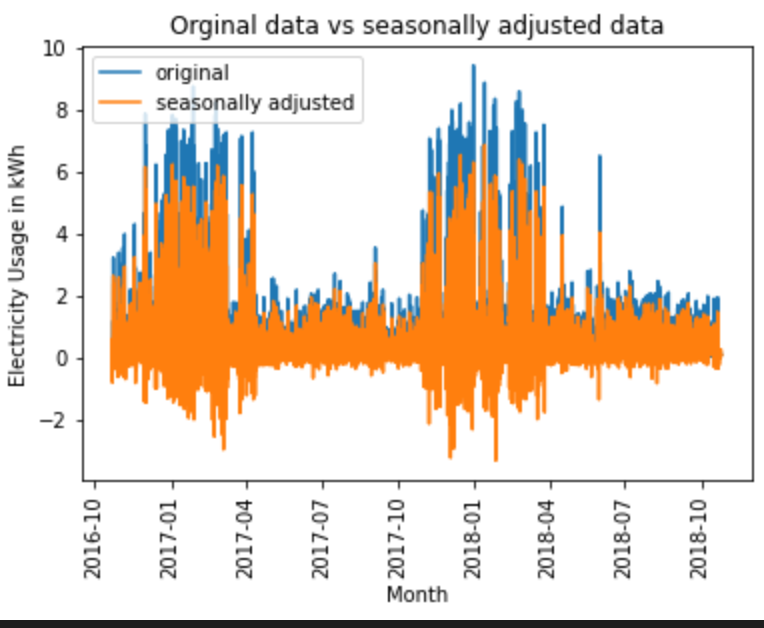


Figure 10 Original Data vs Seasonally adjusted data.

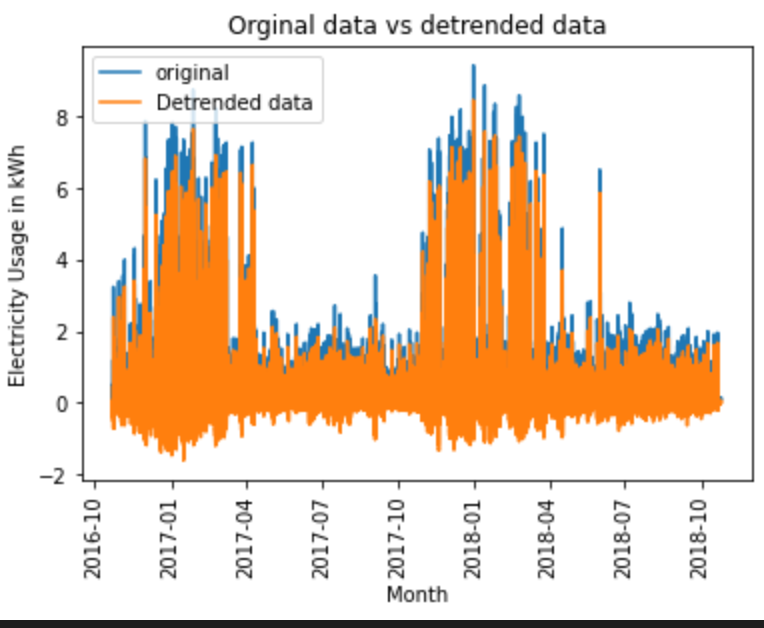
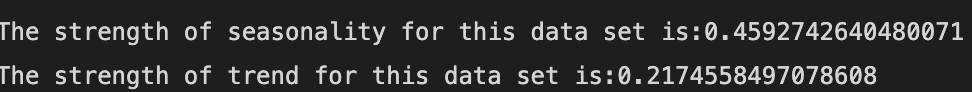


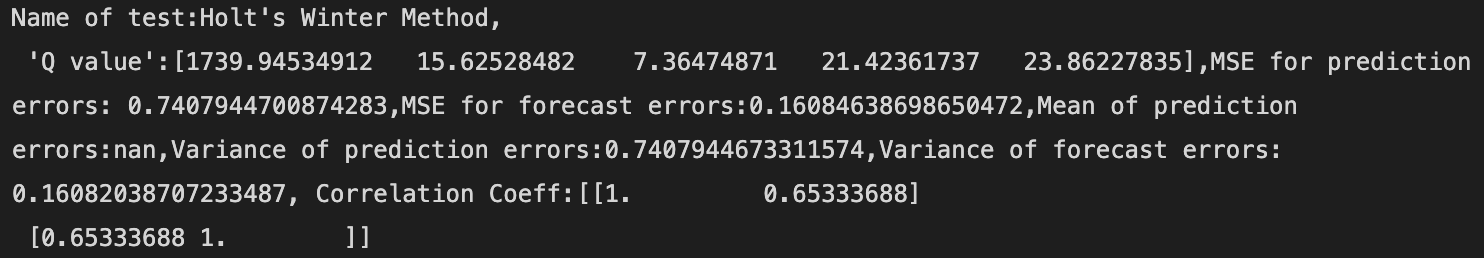
Figure 11 Original data vs Detrended data

From the strength for seasonality and strength for trend, it can be observed that the data is seasonal.



**5 : Holt-Winter Method**

The Mean Squared Error (MSE) for the prediction and the Mean Squared Error (MSE) for the forecast is highlighted for the holt-winter’s method.

****

Form the ACF plot for prediction error it can be seen that the error is white.

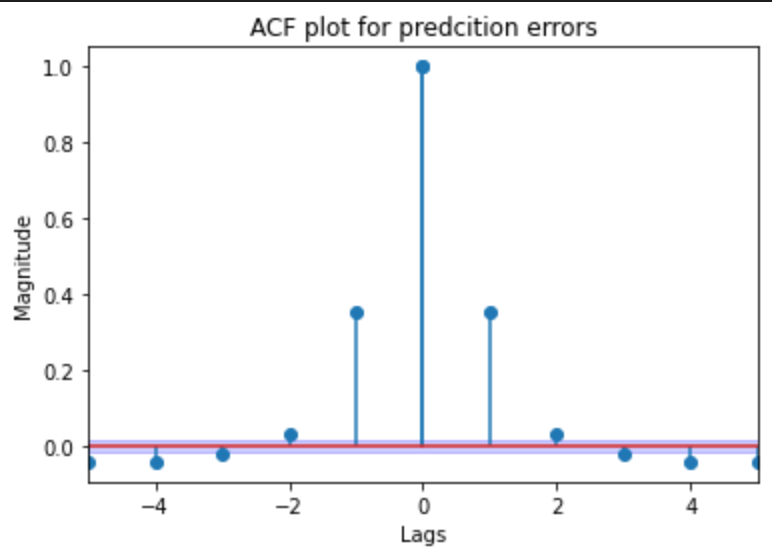


Figure 12 ACF for prediction error (Holt Winters)

Form plot it can be seen that the forecast is not very accurate.

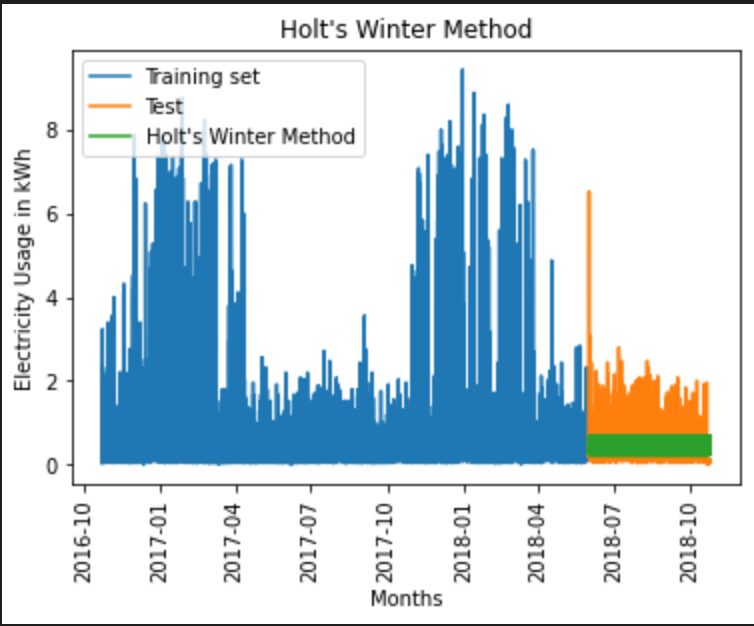
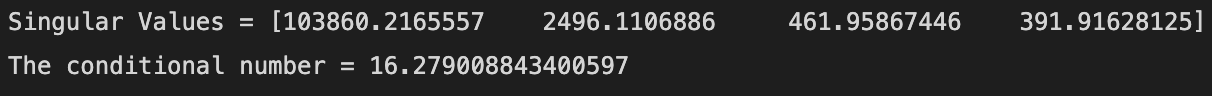
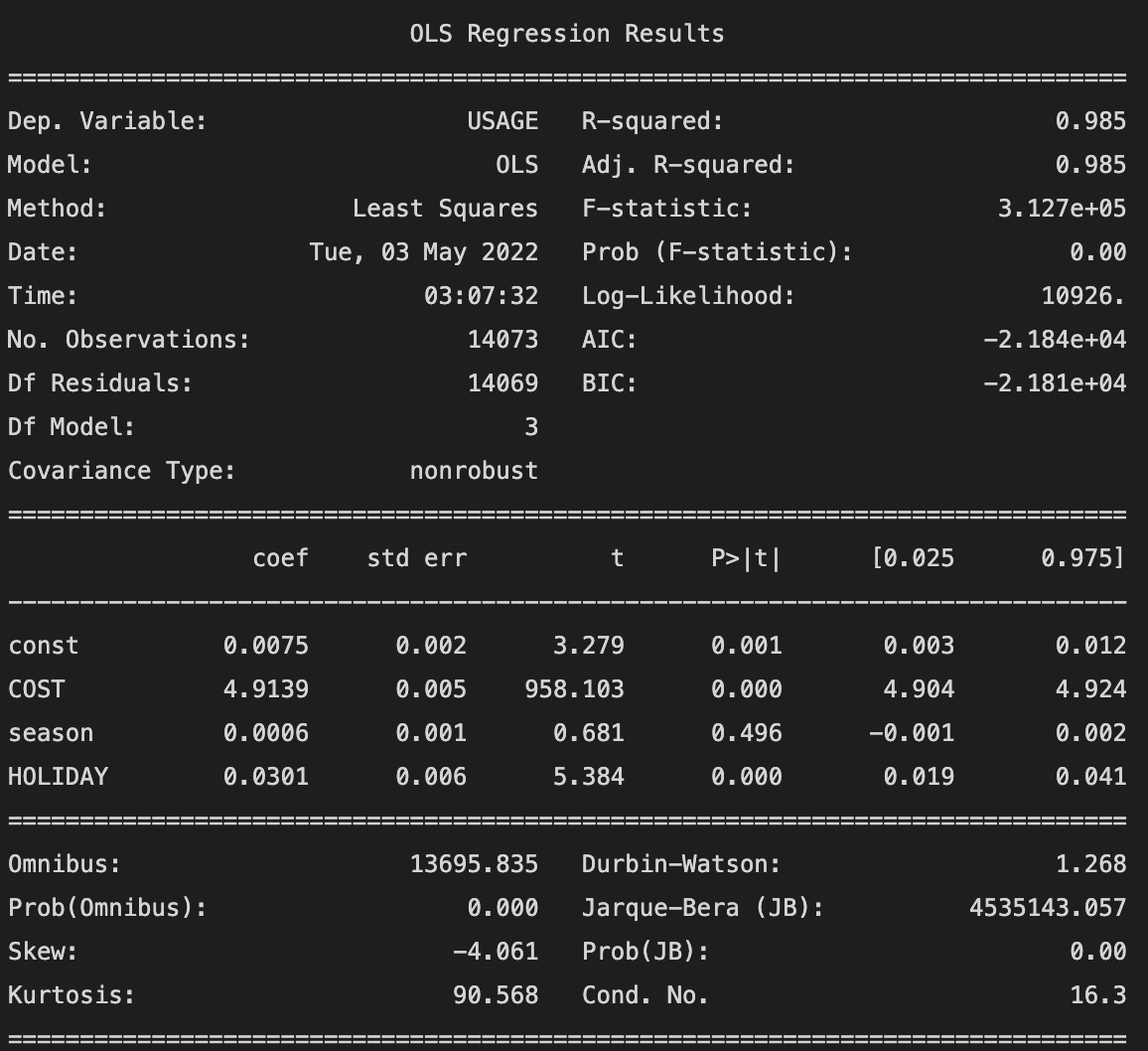


Figure 13 Holt Winters method

**6 : Feature Selection**

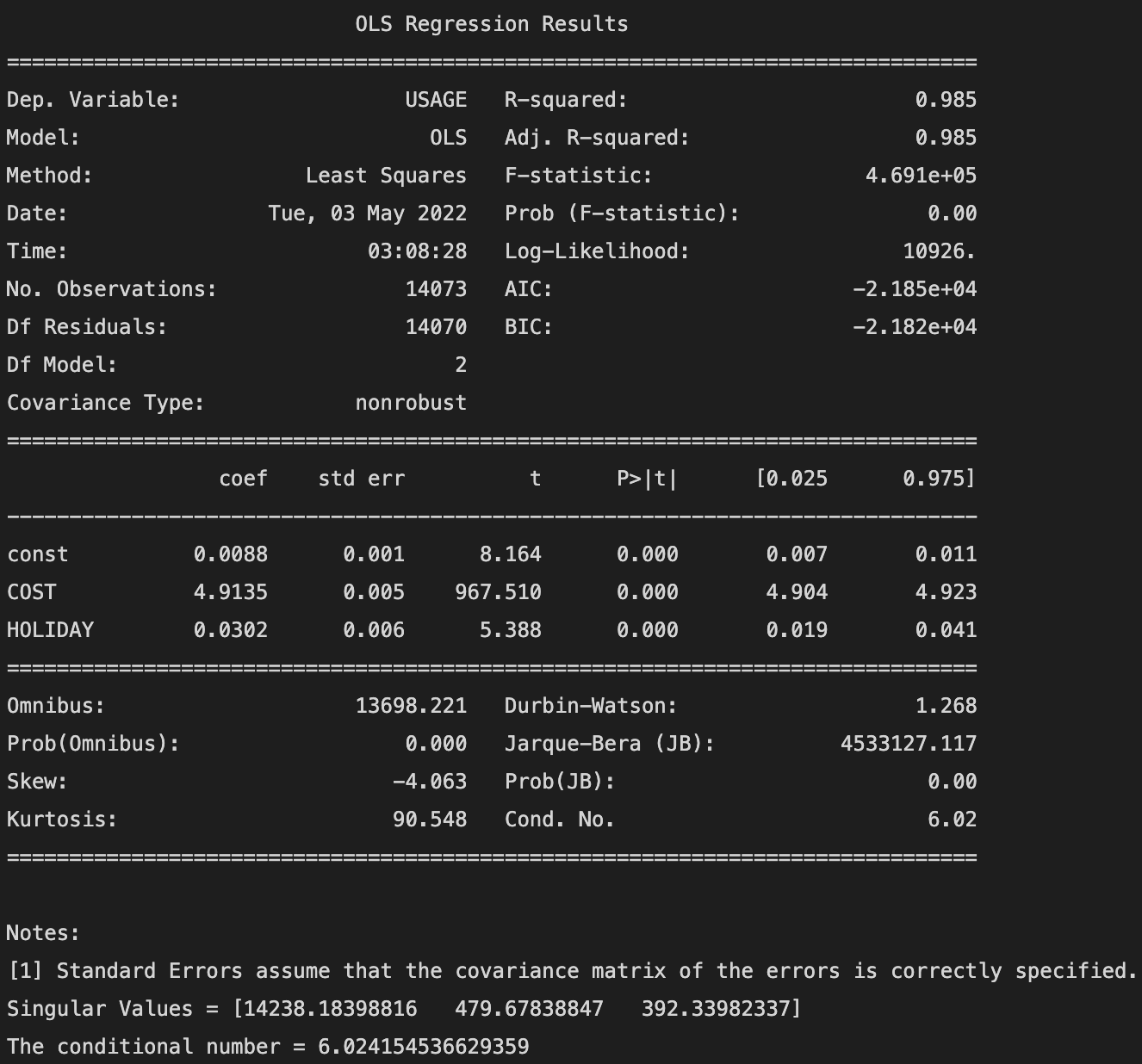
****

All the singular values are greater than zero which does not indicate any collinearity. The Conditional number is also very small which indicates not very strong collinearity.



From the results of the above model, it can be seen that the only season has p value greater than 0.05. Hence the feature needs to be dropped.

After dropping the feature “season”, there is an improvement in the conditional number. Since all the p values are less than 0.05, call the features are not zero.

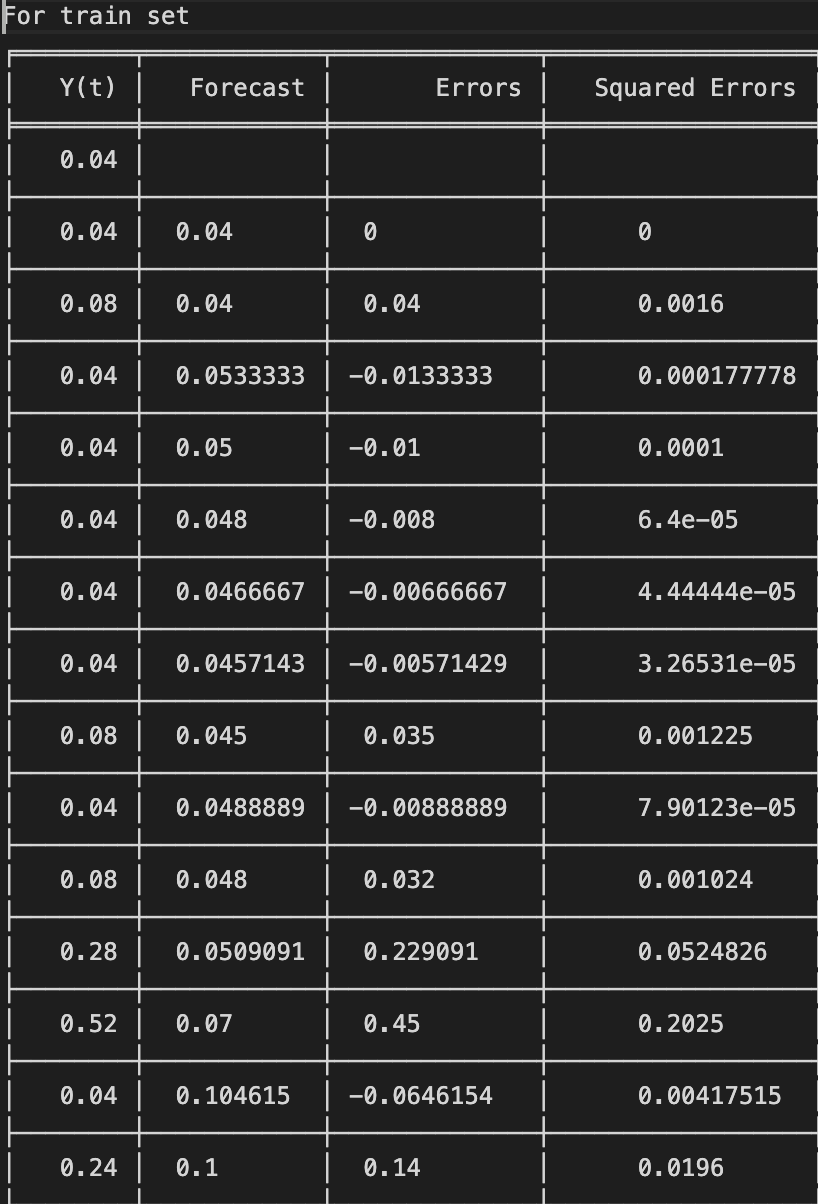
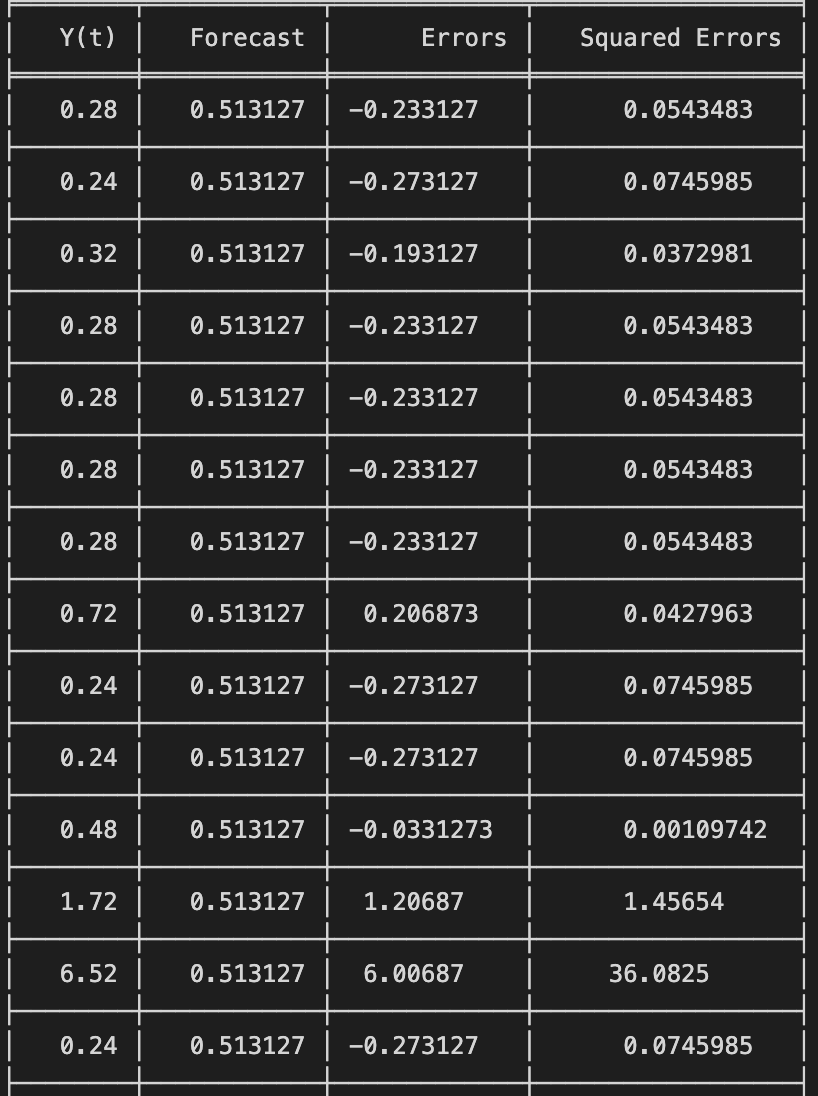


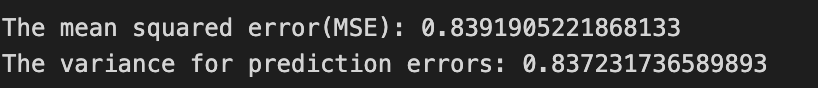
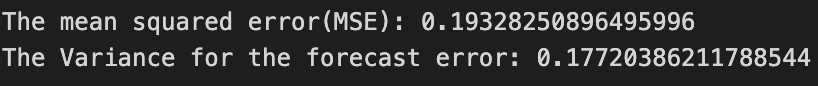
The final features selected for selected for this model are cost and holiday.

**7 : Base Models**

Average method



The average method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

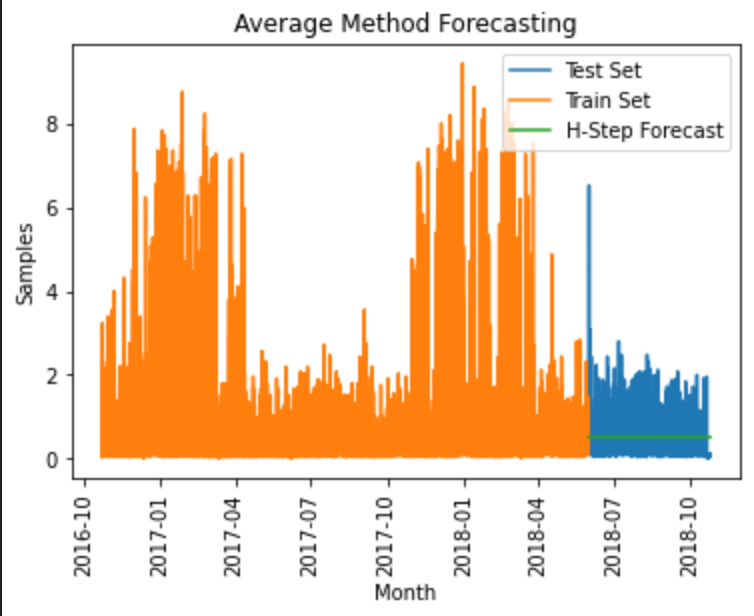
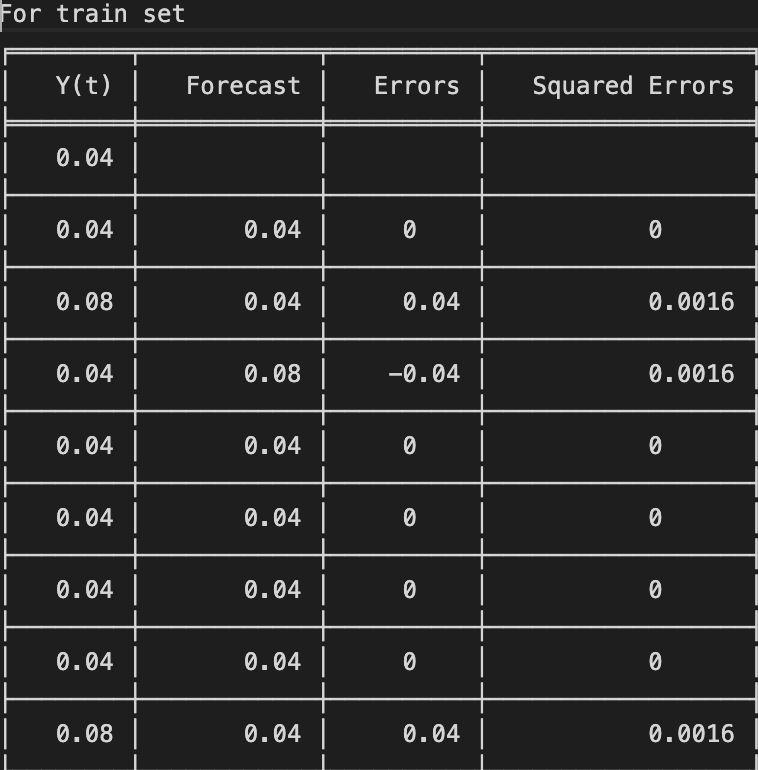
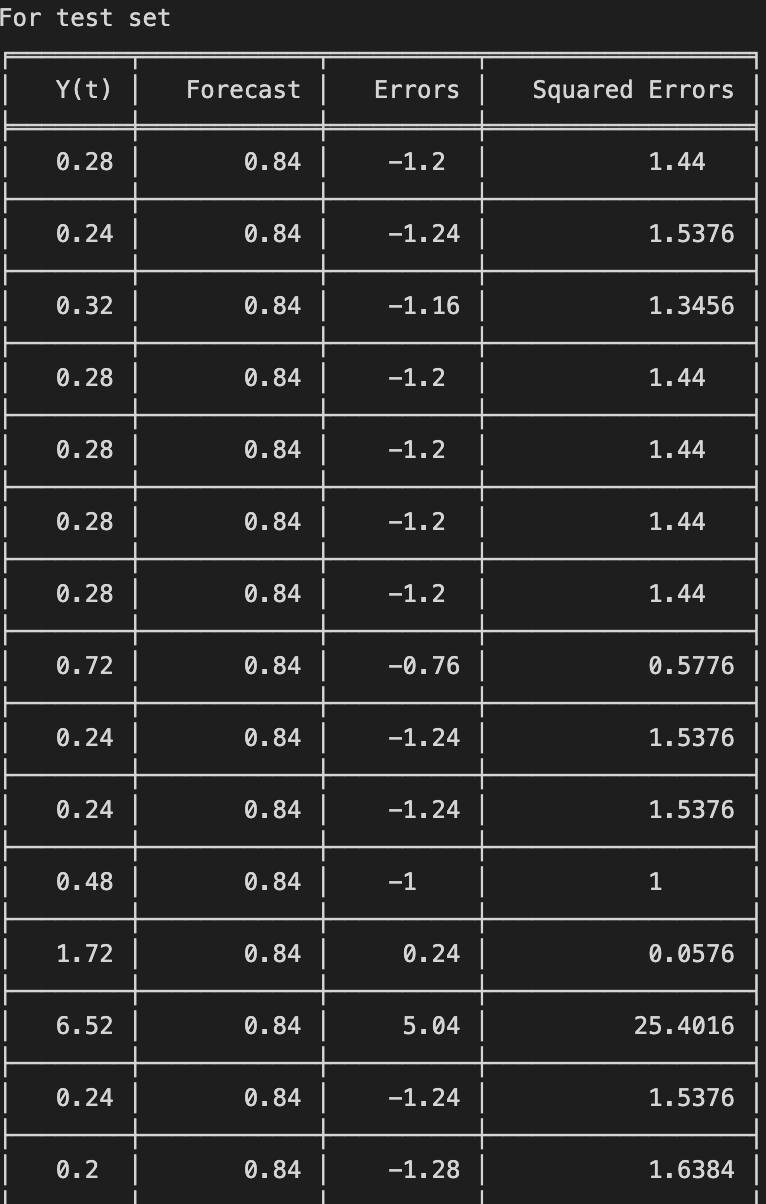
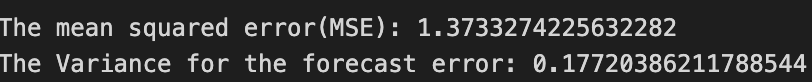
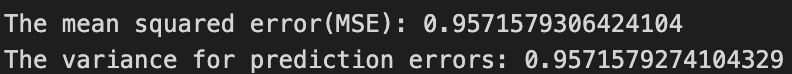


Figure 14 Average method Forecasting

Naïve Method



The naive method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

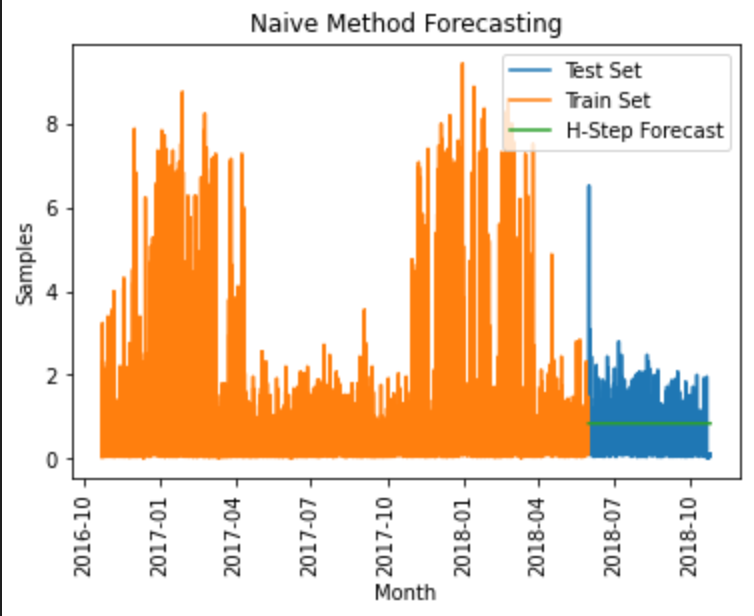
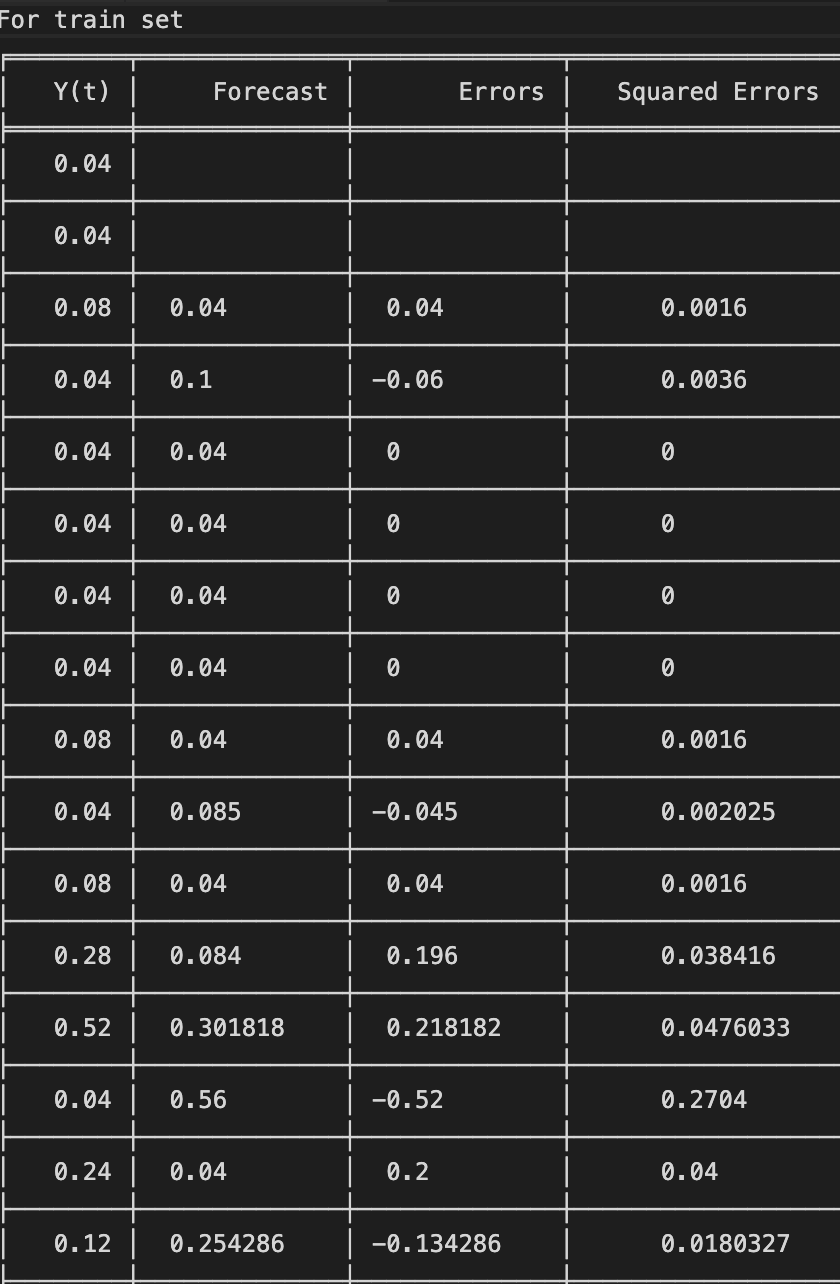
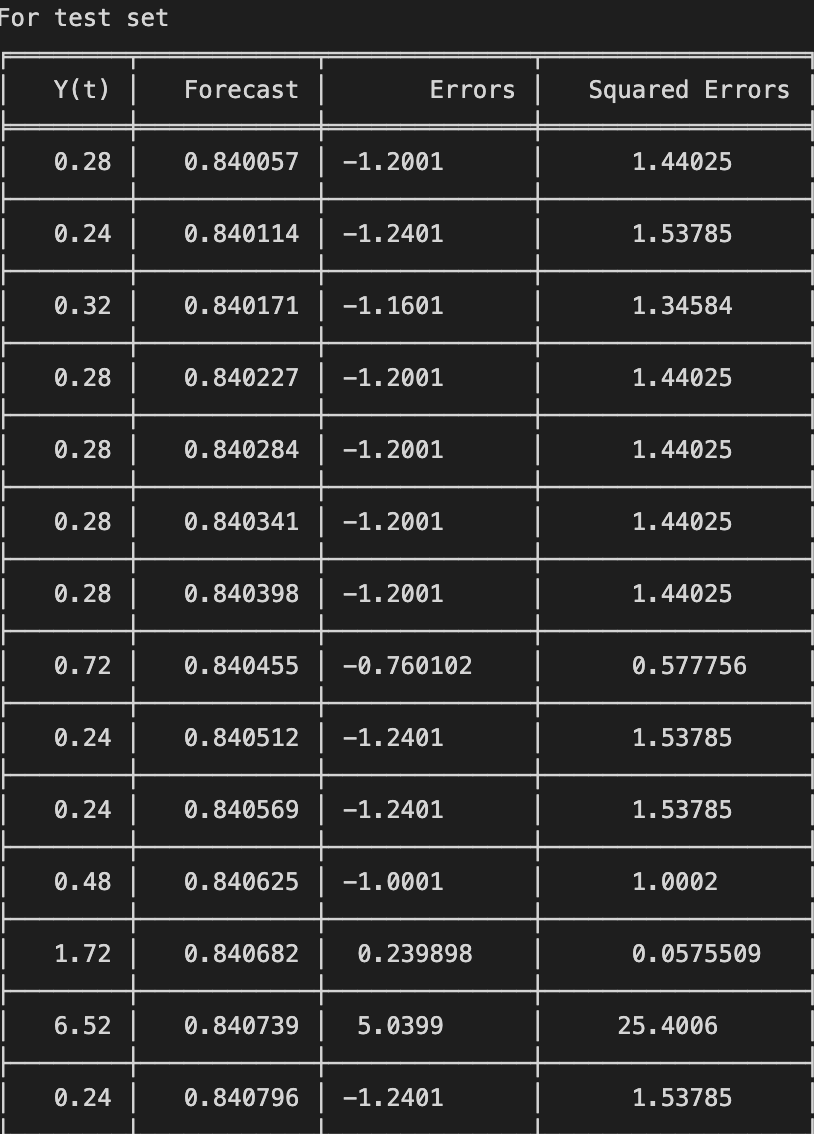
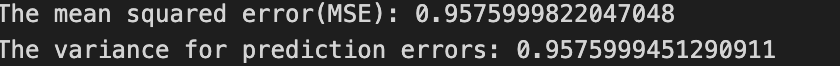
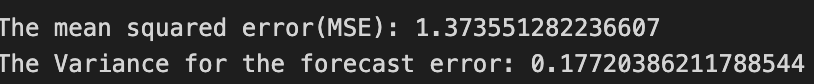


Figure 15 Naive method forecasting

Drift method

The Drift method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

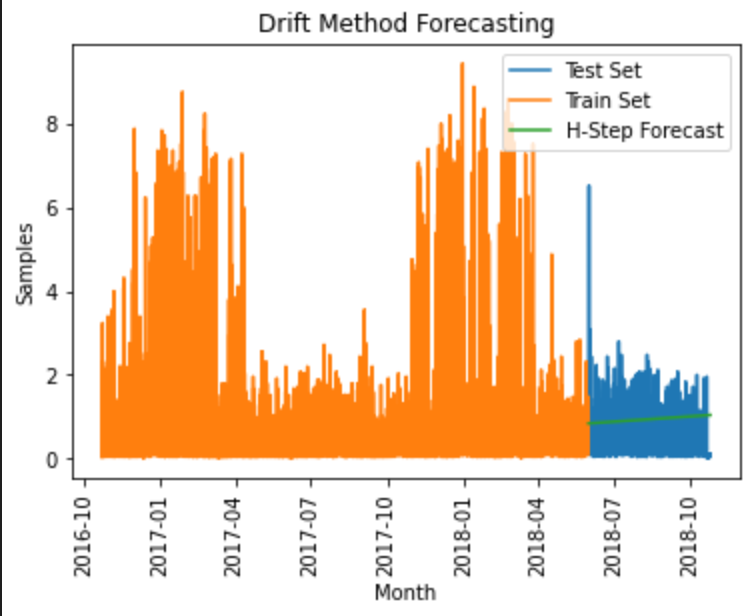
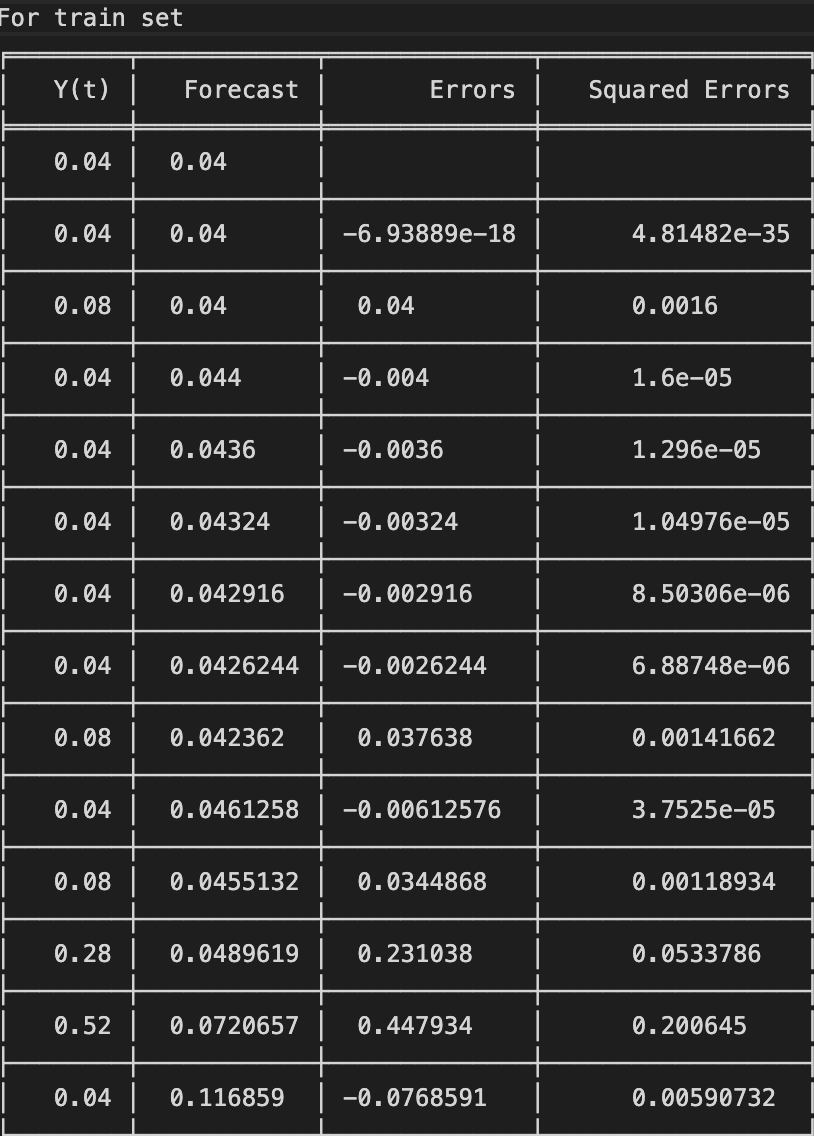
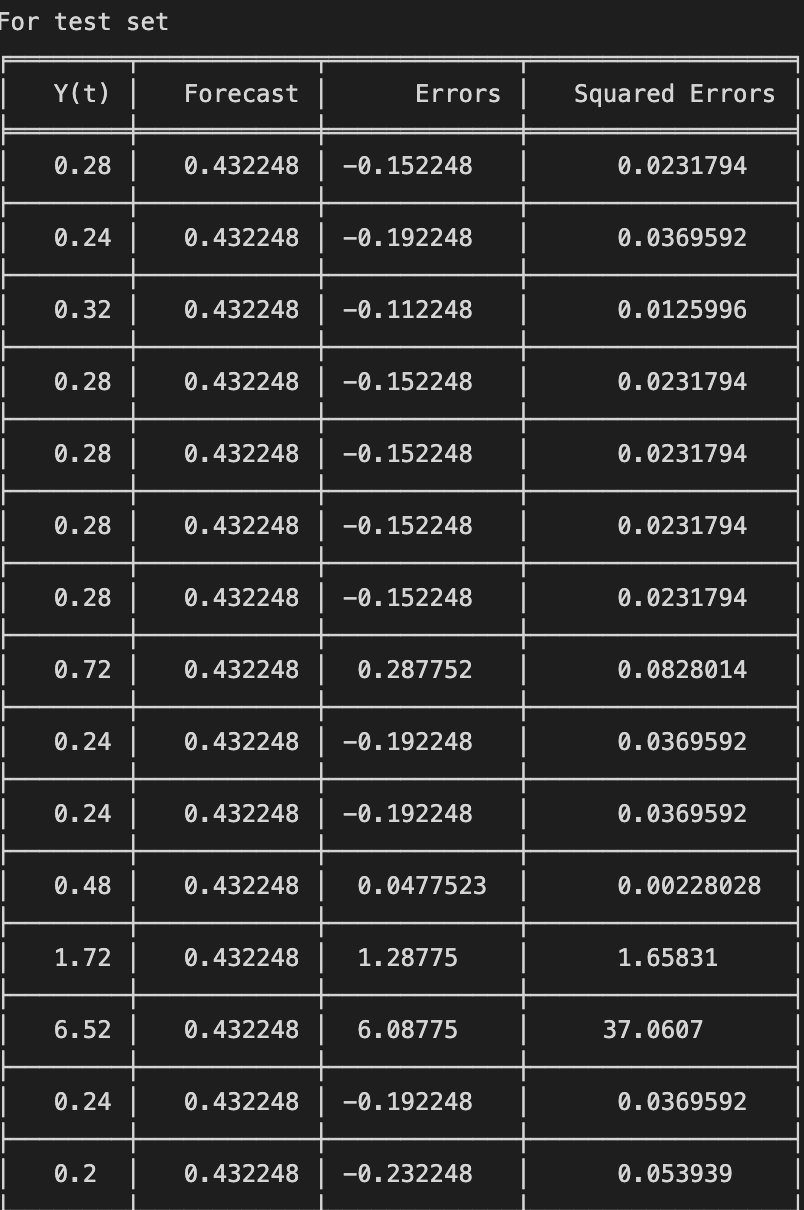
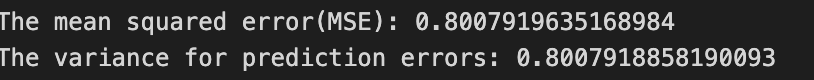
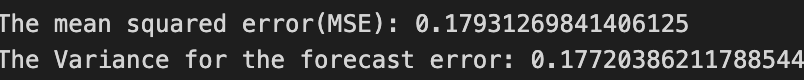


Figure 16 Drift method forecasting

Simple & Exponential Smoothing Method (alpha=0.1)

The SES method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

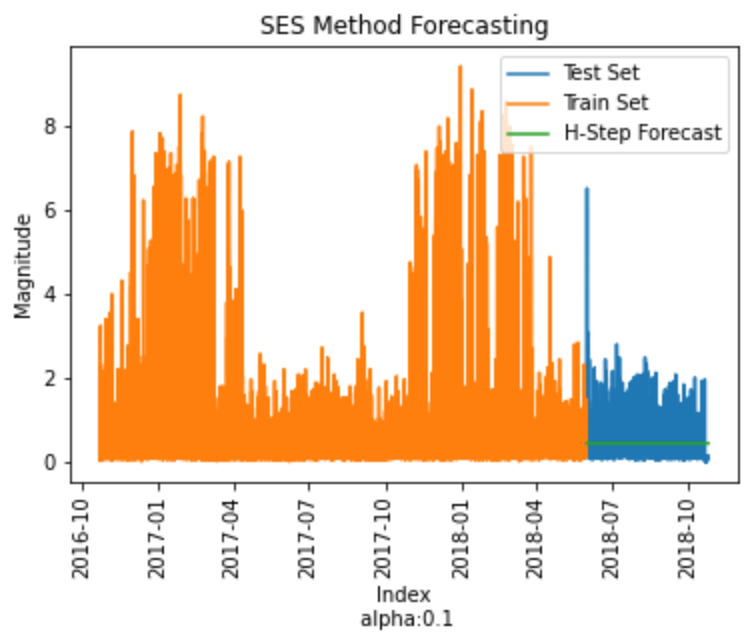
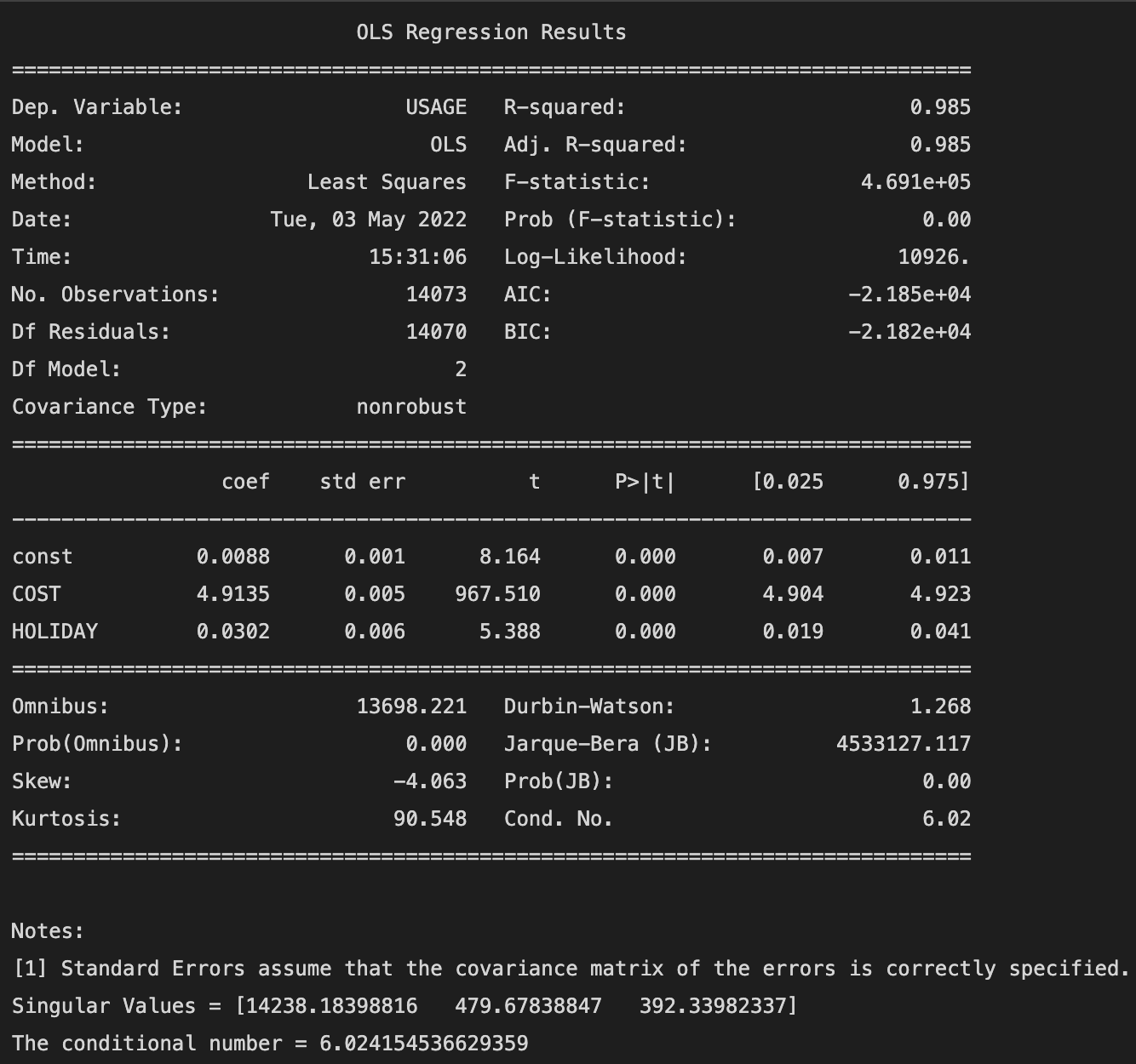
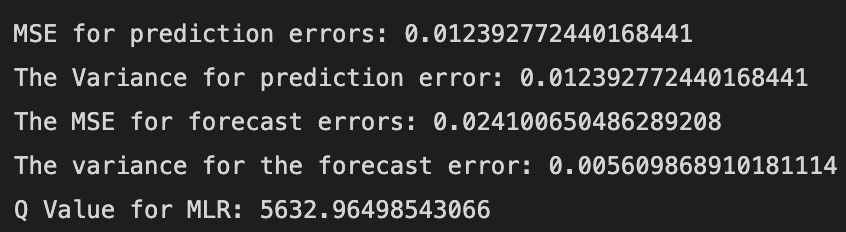


Figure 17 SES method forecasting

**8 : Multiple Linear Regression**



The MLR is perfroming very well. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.



**9 : ARMA, ARIMA & SARIMA**

From the GPAC table it can be seen that the ARMA model has the order of ARMA(2,0).

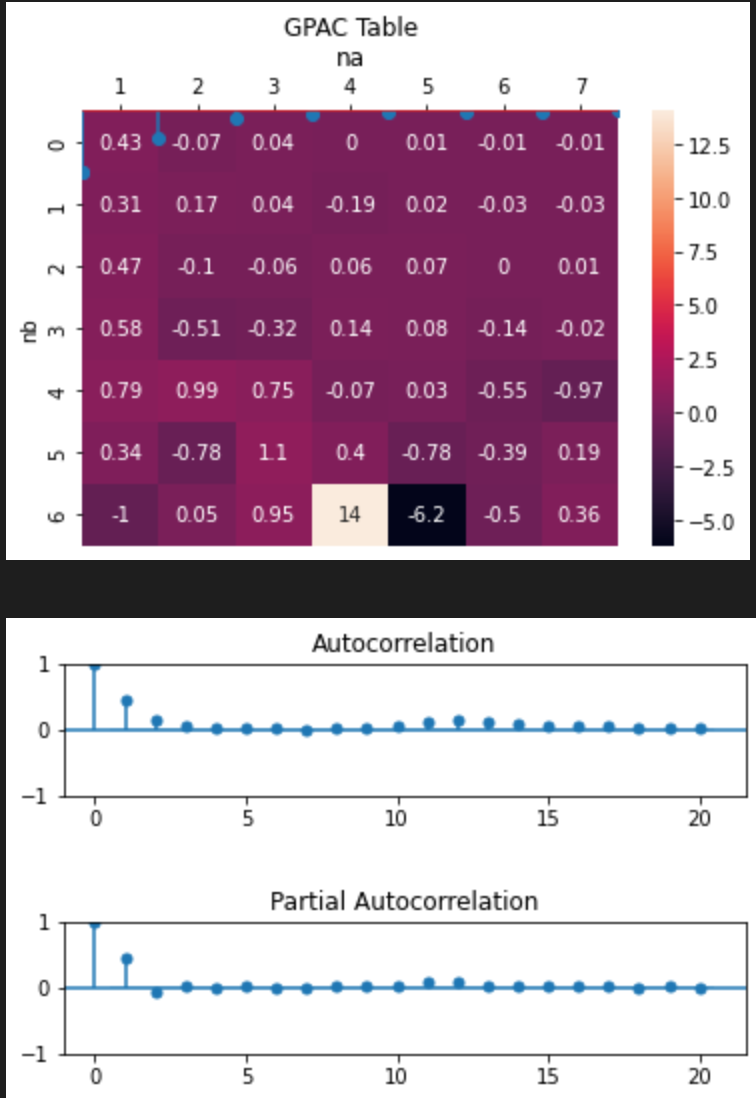


Figure 18 GPAC

From the ACF & PACF after differencing for 12 intervals it is observed that SARIMA(0,1,1)12  and ARIMA(0,12,1) are the appropriate models.

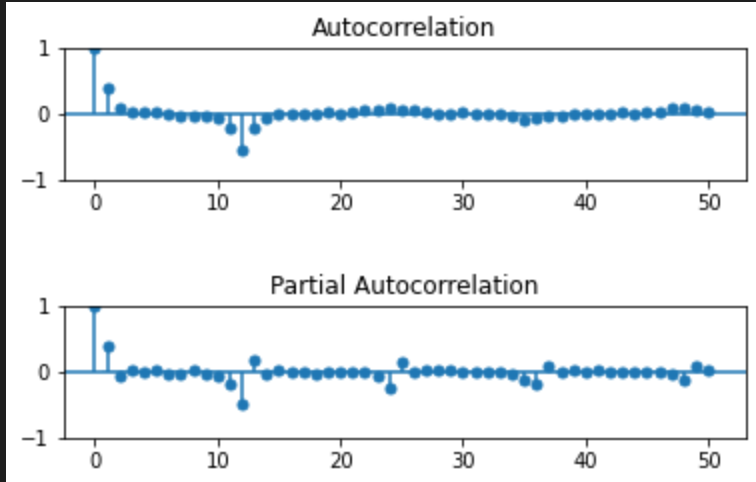


Figure 19 ACF PACF for data after differencing

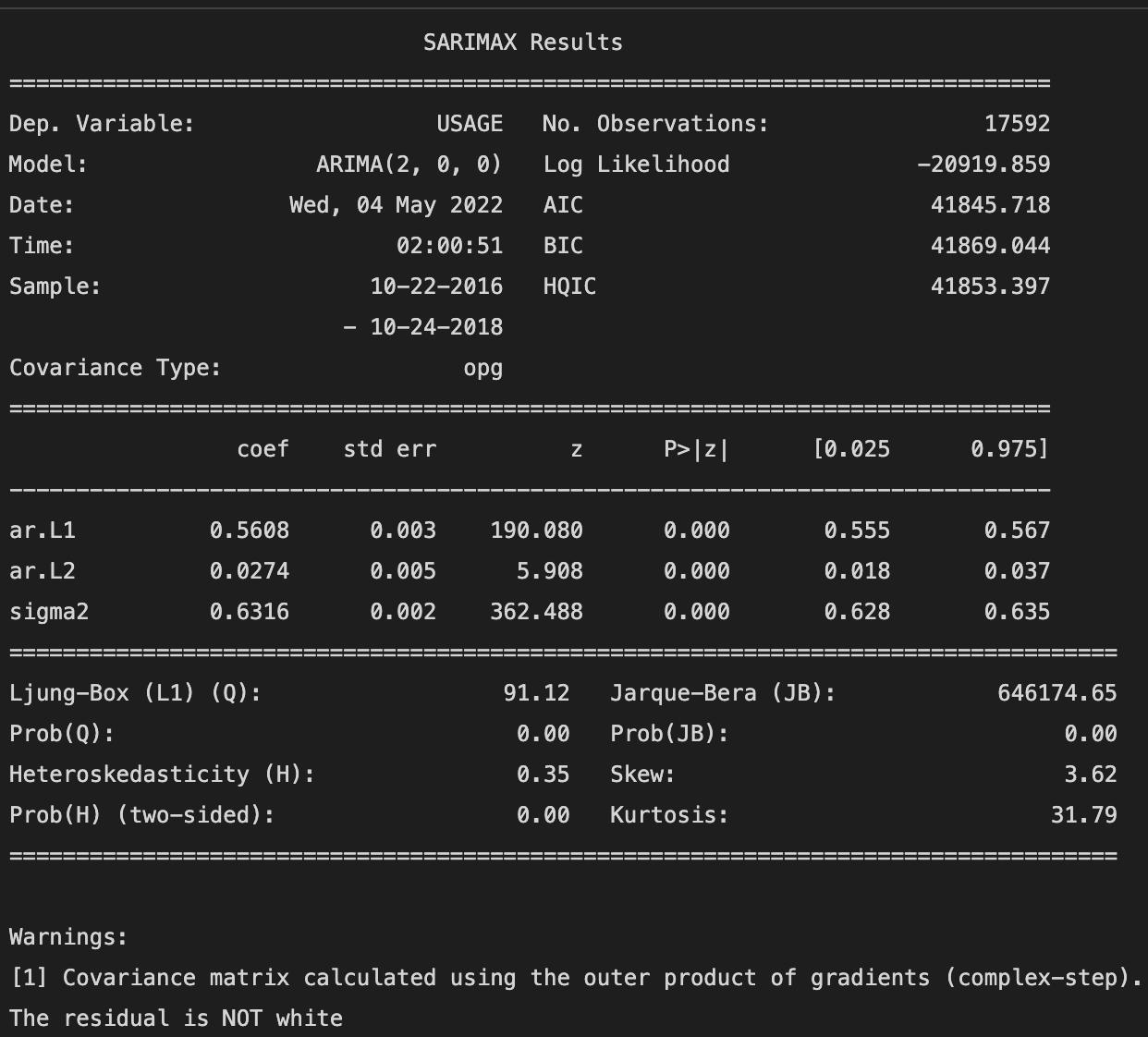
**10 : Levenberg Marquardt algorithm**

From the results the AR parameter are -0.5608 and -0.0274 and are well within the confidence intervals and the p-values are also less than 0.05 which mean all the values are not zero and the coefficients are valid.

Since it’s an AR process only there’s no zero pole cancellations.

As the residual is not white this model is a biased estimator.

**ARMA(2,0)**

****

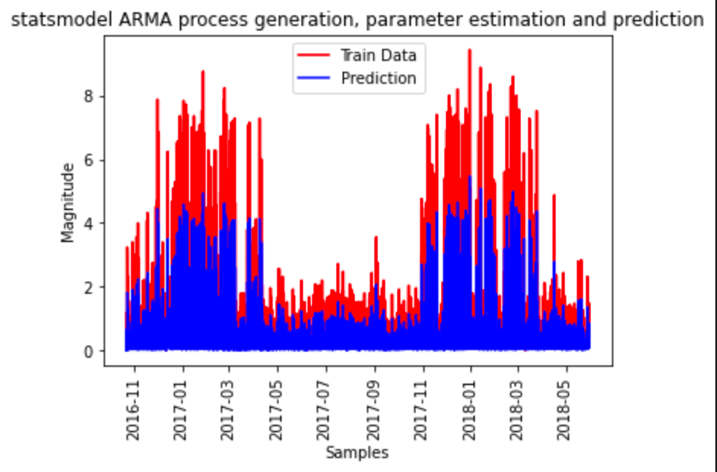
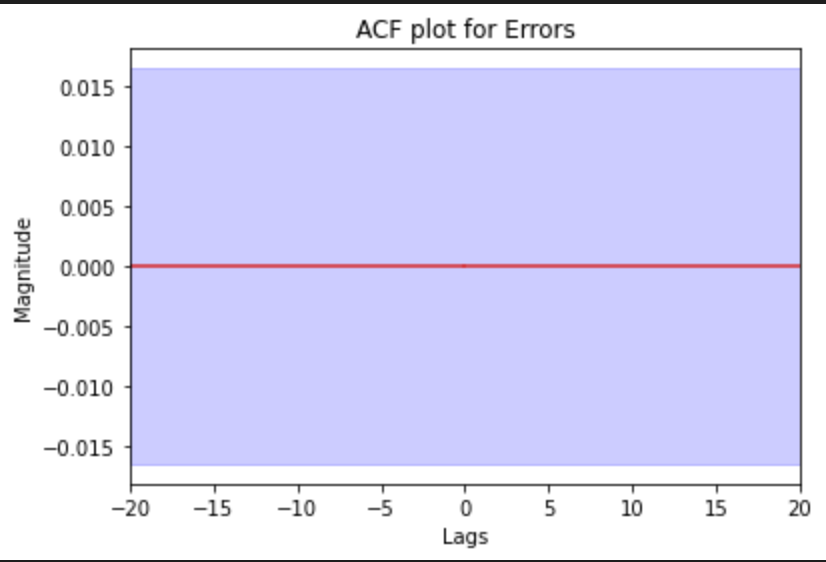
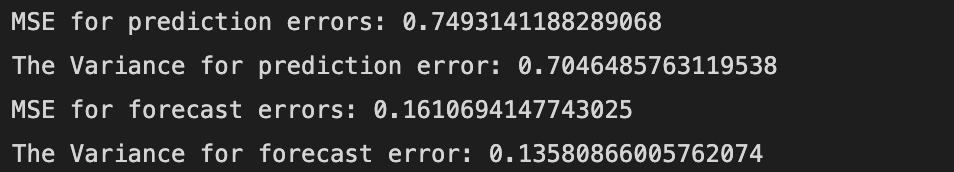
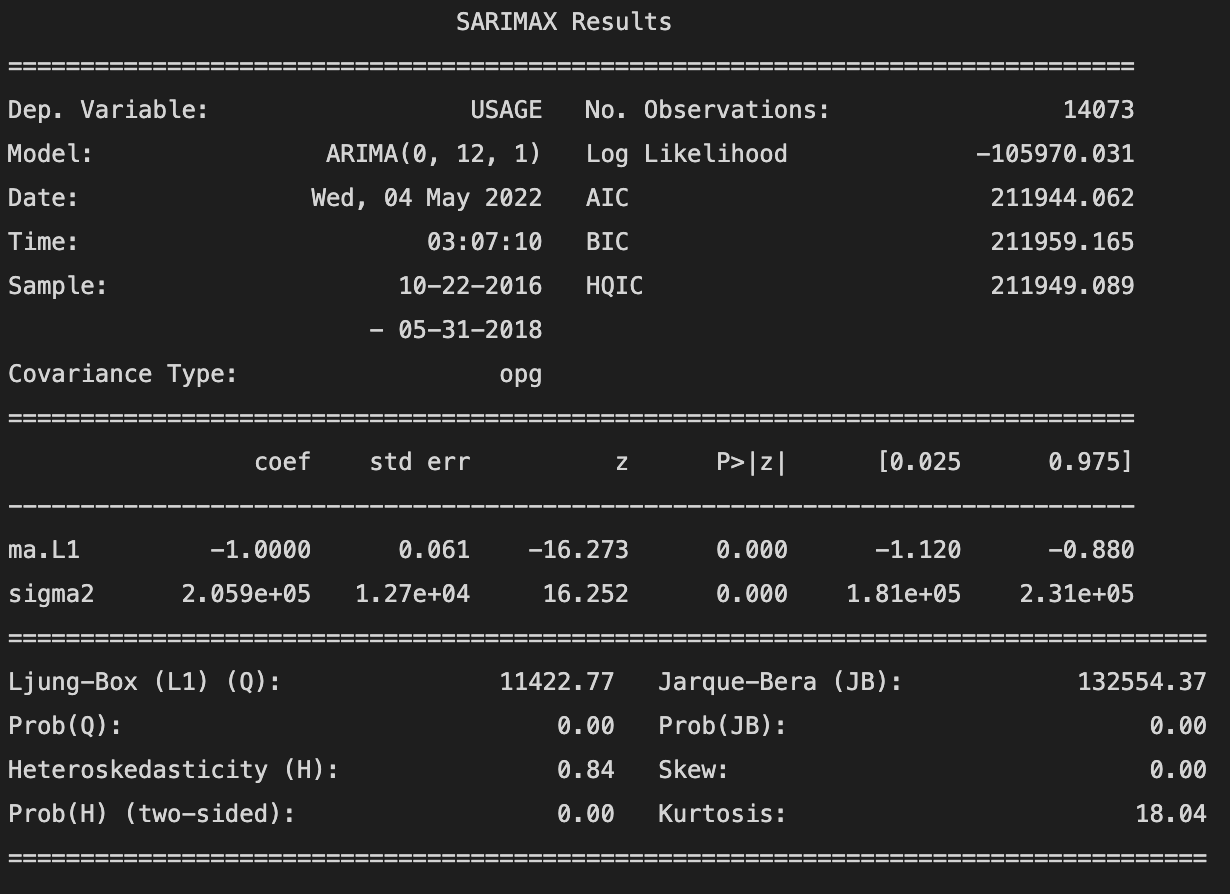
****

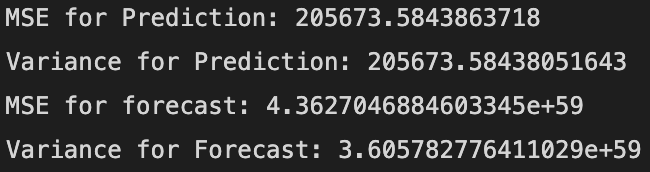
Figure 21 Train and prediction for ARMA(2,0)

Figure 20 ACF for residuals (ARMA)



**ARIMA(0,12,1)**





From the plot below we can see that the residuals are not white and hence this is a biased estimator. The MSE for the prediction and the forecast is also very high which indicated the model is not performing very well.

Hence ARIMA(0.12,1) cannot be used for forecasting the electricity usage.

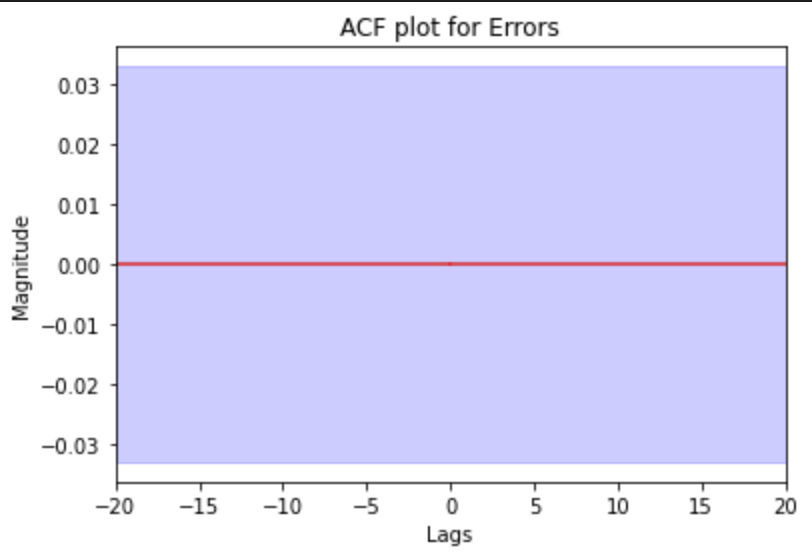


Figure 22 ACF for ARIMA

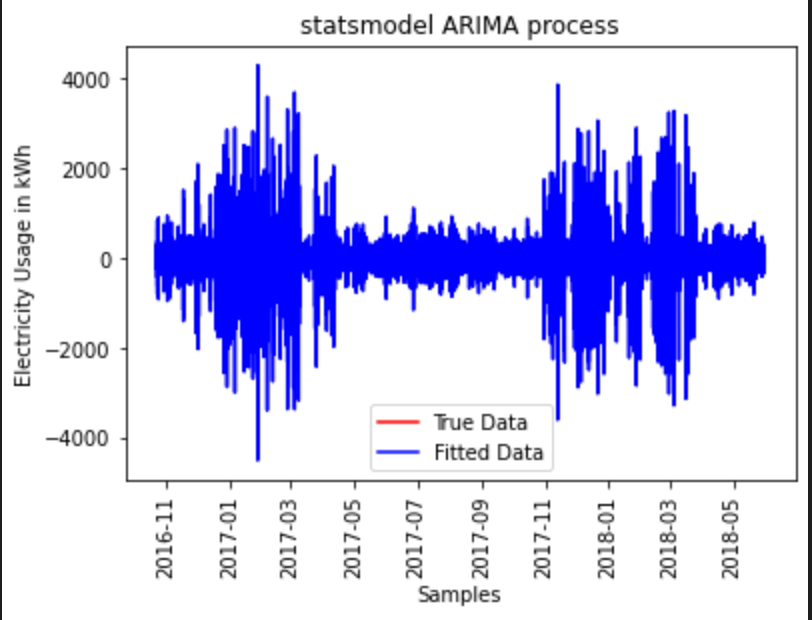
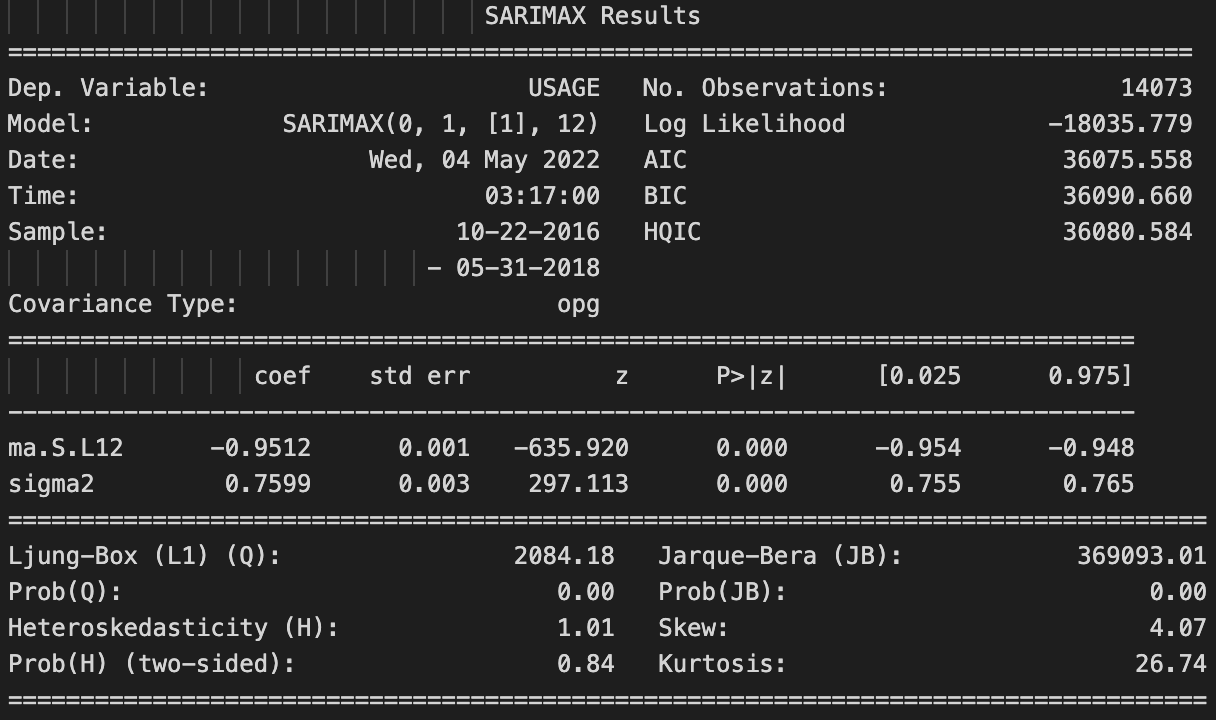
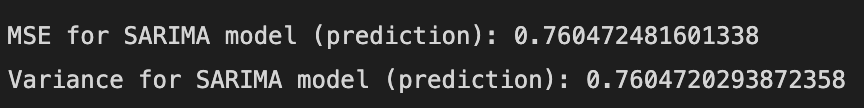
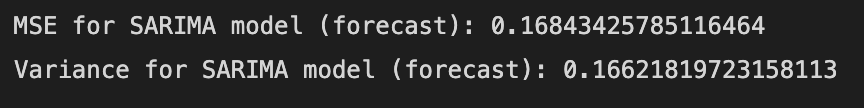


Figure 23 ARIMA Train vs Prediction.

**SARIMA**







The SARIMA model is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

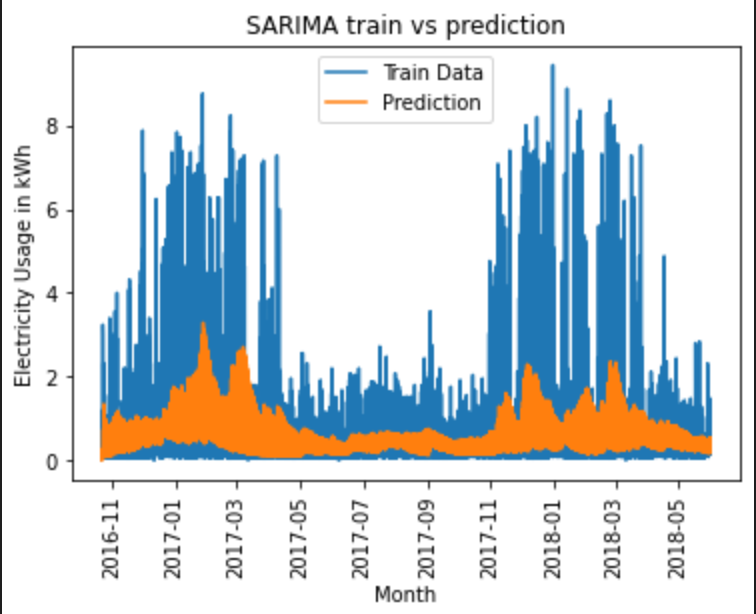


Figure 24 SARIMA train vs prediction

*Table 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **MSE Prediction** | **MSE**  **Forecast** | **Variance**  **Prediction** | **Variance**  **Forecast** |
| Average | 0.8391 | 0.1932 | 0.8372 | 0.1772 |
| Naïve | 0.9571 | 0.1373 | 0.9571 | 0.1772 |
| Drift | 0.9575 | 1.3735 | 0.9575 | 0.1772 |
| Holt Winters | 0.7407 | 0.1608 | 0.7407 | 0.1608 |
| SES | 0.8007 | 0.1793 | 0.8007 | 0.1772 |
| MLR | 0.0123 | 0.0241 | 0.0123 | 0.0056 |
| ARMA | 0.7493 | 0.1610 | 0.7046 | 0.1358 |
| ARIMA | 205673.58 | 4.3627 X 1059 | 205673.58 | 3.6057 |
| SARIMA | 0.7604 | 0.1684 | 0.7604 | 0.1662 |

**11 : Best Model Selection**

Form table 1 it can be seen that multiple linear regression has the least mean squared error (MSE) and variance for prediction and the forecast. Hence the best model for this data is the multiple linear regression model (MLR).

**12 : h-step prediction**



Figure 25 h-step prediction for MLR

From the above figure it can be seen that the forecast is exactly in line with the test set which means the model is forecasting the values very well.

**13 : Conclusion**

The multiple regression model has given the best results for prediction as well as the forecast. It had the least MSE than any other model used for prediction. The plot also shows that the forecasts are exactly in line with the test set.

A better model than this could be the LSTM model which uses deep learning and could’ve given better results.

**14: References**

Data: <https://www.kaggle.com/jaganadhg/house-hold-energy-data>

**15 : Appendix**

#%%

from tracemalloc import start

from scipy.stats import chi2

from cProfile import label

from multiprocessing.spawn import import\_main\_path

from unicodedata import category, name

from importlib\_metadata import distribution

import pandas as pd

# from Lab4 import X\_test, X\_train

import numpy as np

from pyparsing import col

import toolbox as tx

import matplotlib.pyplot as plt

import seaborn as sns

from datetime import datetime

from sklearn.model\_selection import train\_test\_split

from pandas.tseries.holiday import USFederalHolidayCalendar as calendar

import math

from statsmodels.tsa.seasonal import STL

import statsmodels.tsa.holtwinters as ets

import statsmodels.api as sm

from scipy import signal

# %%

def difference(dataset, interval =1):

diff = []

for i in range(interval, len(dataset)):

value = dataset[i]- dataset[i-interval]

diff.append(value)

return diff

def log\_trans(column):

lg= []

for i in range(len(column)):

value = math.log(column[i])

lg.append(value)

return lg

#%%

#%%

df = pd.read\_csv("/Users/atharvah/GWU/Untitled/personal/Time\_Series/Labs/D202.csv")

df.DATE = pd.to\_datetime(df.DATE + ' ' + df["END TIME"])

df = df.drop(columns=["START TIME","END TIME"])

df = df.set\_index(df.DATE)

# 1: spring,2: summer,3: fall,4: winter

df["season"]= df.index.month%12 // 3 + 1

#%%

# Citation: JAGANADH GOPINADHAN

# Check if holiday or not

cal = calendar()

holidays = cal.holidays(start = df.index.min(), end = df.index.max())

df["HOLIDAY"] = df.index.strftime('%Y-%m-%d').isin(holidays.strftime('%Y-%m-%d'))

df["HOLIDAY"] = pd.Categorical(df["HOLIDAY"])

df.season = pd.Categorical(df.season)

#%%

df.COST = df.COST.str[1:]

df.COST = df.COST.astype(float)

#%%

#%%

# Data Pre-processing

display(df.head())

display(df.tail())

missing\_vals = df.isna().sum()

print("The numeber of Missing Values:", missing\_vals)

#%%

# Dropping column with empty value and irrelevant values

df = df.drop(columns=["NOTES","TYPE","UNITS"])

# Displaying the Shape of the dataframe

print(f"There are {len(df.index)} rows and {len(df.columns)} columns in the dataset.")

display(df.head())

#%%

# Citation : SANKET SHARMA

# Hourly

hourly\_usage\_df = df.resample("1H", on="DATE").sum()

display(hourly\_usage\_df.head())

# Daily

daily\_usage\_df = hourly\_usage\_df.resample("1D").sum()

display(daily\_usage\_df.head())

# Monthly

monthly\_usage\_df = daily\_usage\_df.resample("1M").sum()

display(monthly\_usage\_df.head())

#%%

monthly\_usage\_df["per\_unit\_cost"] = monthly\_usage\_df["COST"]/monthly\_usage\_df["USAGE"]

monthly\_usage\_df["year"] = monthly\_usage\_df.index.year.astype(str)

monthly\_usage\_df["month\_name"] = monthly\_usage\_df.index.month\_name().str[:3]

monthly\_usage\_df["usage\_month"] = monthly\_usage\_df["month\_name"] + "-" + monthly\_usage\_df["year"]

monthly\_usage\_df = monthly\_usage\_df.drop(columns=["year", "month\_name"], errors="ignore")

hourly\_usage\_df["year"] = hourly\_usage\_df.index.year.astype(str)

hourly\_usage\_df["month\_name"] = hourly\_usage\_df.index.month\_name().str[:3]

hourly\_usage\_df["usage\_month"] = hourly\_usage\_df["month\_name"] + "-" + hourly\_usage\_df["year"]

hourly\_usage\_df["season"]= hourly\_usage\_df.index.month%12 // 3 + 1

hourly\_usage\_df.season = pd.Categorical(hourly\_usage\_df.season)

hourly\_usage\_df = hourly\_usage\_df.drop(columns=["year", "month\_name"], errors="ignore")

hourly\_usage\_df["HOLIDAY"] = hourly\_usage\_df.index.strftime('%Y-%m-%d').isin(holidays.strftime('%Y-%m-%d')).astype(int)

hourly\_usage\_df["HOLIDAY"] = pd.Categorical(hourly\_usage\_df["HOLIDAY"])

display(hourly\_usage\_df.head())

#%%

tx.ACF\_PACF\_Plot(hourly\_usage\_df.USAGE,25)

#%%

# Target variable vs Time

plt.plot(df.USAGE)

plt.title("Usage in every 15 Minutes")

plt.xlabel("Month")

plt.ylabel("Electricity Usage in KWh")

plt.xticks(rotation = 90)

plt.show()

#%%

# Change in unit rate montly

plt.plot(monthly\_usage\_df.index,monthly\_usage\_df.per\_unit\_cost)

plt.title("Monthly Change in Price per unit")

plt.xlabel("Month")

plt.ylabel("Electricity price/unit ($)")

plt.xticks(rotation = 90)

plt.show()

#%%

# Hourly change in usage of electricity

plt.plot(hourly\_usage\_df.USAGE)

plt.title("Hourly Usage")

plt.xlabel("Month")

plt.ylabel("Electricity Usage in KWh")

plt.xticks(rotation = 90)

plt.show()

#%%

#%%

datatypes = hourly\_usage\_df.dtypes

display(datatypes)

#%%

X\_train, X\_test = train\_test\_split(hourly\_usage\_df,test\_size=0.2,shuffle=False)

# %%

# tx.Cal\_rolling\_mean\_var(hourly\_usage\_df.USAGE)

# Since the rolling mean and the rolling variance turns into a straight line the data is stationary.

# %%

tx.ADF\_Cal(hourly\_usage\_df.USAGE)

# We reject the null hypothesis hence the data is staitonary according to ADF test

# %%

tx.kpss\_test(hourly\_usage\_df.USAGE)

# We failt reject the null hypothesis hence the data is staitonary according to kpss test

#%%

# %%

tx.stem\_plot(hourly\_usage\_df.USAGE,20,name="Hourly electricity usage")

# %%

sns.pairplot(hourly\_usage\_df,kind="kde").set(title='Hourly Usage')

# %%

df1 = hourly\_usage\_df.corr()

sns.heatmap(df1,annot=True).set(title = "Correlation matrix for hourly Usage ")

# %%

STL = STL(hourly\_usage\_df.USAGE)

res = STL.fit()

fig = res.plot()

plt.show()

#%%

S= res.seasonal

T = res.trend

R = res.resid

#%%

plt.plot(R,label="Residual")

plt.plot(S,label="Seasonal")

plt.plot(T,label="Trend")

plt.legend(loc = "best")

plt.xlabel("months")

plt.ylabel("Electricity Usage in kWh")

plt.title("Plot for seasonality, trend and residual")

plt.xticks(rotation = 90)

plt.show()

# %%

adj\_seasonal = hourly\_usage\_df.USAGE-S

plt.plot(hourly\_usage\_df.USAGE,label="original")

plt.plot(adj\_seasonal,label="seasonally adjusted")

plt.xlabel("Month")

plt.ylabel("Electricity Usage in kWh")

plt.title("Orginal data vs seasonally adjusted data")

plt.legend(loc = "best")

plt.xticks(rotation = 90)

plt.show

# %%

F = np.maximum(0,1-np.var(np.array(R))/np.var(np.array(S)+np.array(R)))

print(f"The strength of seasonality for this data set is:{F}")

F1 = np.maximum(0,1-np.var(np.array(R))/np.var(np.array(T)+np.array(R)))

print(f"The strength of trend for this data set is:{F1}")

#%%

detrended\_pass= hourly\_usage\_df.USAGE-T

plt.plot(hourly\_usage\_df.USAGE,label="original")

plt.plot(detrended\_pass,label="Detrended data")

plt.xlabel("Month")

plt.ylabel("Electricity Usage in kWh")

plt.title("Orginal data vs detrended data")

plt.legend(loc = "best")

plt.xticks(rotation = 90)

plt.show

# %%

holtt = ets.ExponentialSmoothing(X\_train.USAGE,seasonal ="add",damped\_trend = False).fit()

holtf = holtt.forecast(steps=len(X\_test.USAGE))

holtf = pd.DataFrame(holtf).set\_index(X\_test.index)

#%%

holtt.summary()

#%%

error\_hw = X\_test.USAGE - holtf[0]

er\_sqd\_hw = error\_hw\*\*2

pred\_error\_hw = X\_train.USAGE-holtt.fittedvalues

rk= tx.stem\_plot(pred\_error\_hw,5,name="predcition errors")

qv = (len(X\_train))\*(np.array(rk[1:])\*\*2)

af = print(f"Name of test:Holt's Winter Method,\n 'Q value':{qv},MSE for prediction errors: {np.mean((X\_train.USAGE-holtt.fittedvalues)\*\*2)},MSE for forecast errors:{np.mean(er\_sqd\_hw)},Variance of prediction errors:{np.var(X\_train.USAGE-holtt.fittedvalues)},Variance of forecast errors: {np.var(error\_hw)}, Correlation Coeff:{np.corrcoef(er\_sqd\_hw,X\_test.USAGE)}")

# tab = tab.append(af,ignore\_index = True)

fig, ax = plt.subplots()

ax.plot(X\_train.USAGE,label = "Training set")

ax.plot(X\_test.USAGE,label ="Test")

ax.plot(holtf[0],label = "Holt's Winter Method")

ax.set\_xlabel("Months")

ax.set\_ylabel("Electricity Usage in kWh")

ax.set\_title("Holt's Winter Method")

plt.xticks(rotation = 90)

ax.legend()

plt.show()

# %%

# Average forecasting

tx.average\_forecasting(hourly\_usage\_df.USAGE)

# %%

# Naive forecasting

tx.N\_forecasting(hourly\_usage\_df.USAGE)

# %%

# SES forecasting

tx.ses\_forecasting(hourly\_usage\_df.USAGE,alpha=0.1)

# %%

# Drift method forecasting

tx.d\_forecasting(hourly\_usage\_df.USAGE)

# %%

X= hourly\_usage\_df.loc[:,hourly\_usage\_df.columns!='USAGE']

X= X.drop(columns="usage\_month")

Y = hourly\_usage\_df.loc[:,hourly\_usage\_df.columns=='USAGE']

X = sm.add\_constant(X)

#%%

X\_tr , X\_ts,Y\_tr, Y\_ts = train\_test\_split(X,Y,test\_size=0.2, shuffle=False)

#%%

Xm = np.array(X\_tr)

Ym = np.array(Y\_tr)

h = np.matmul(Xm.T,Xm)

# %%

s, d, v = np.linalg.svd(h)

# %%

print("Singular Values =",d)

print("The conditional number =",np.linalg.cond(Xm))

# %%

# Coefficients from MLR

model = sm.OLS(Y\_tr,X\_tr).fit()

print(model.summary())

# %%

X\_tr = X\_tr.drop(columns="season")

# %%

model = sm.OLS(Y\_tr,X\_tr).fit()

print(model.summary())

Xm = np.array(X\_tr)

Ym = np.array(Y\_tr)

h = np.matmul(Xm.T,Xm)

s, d, v = np.linalg.svd(h)

print("Singular Values =",d)

print("The conditional number =",np.linalg.cond(Xm))

# %%

X\_ts = X\_ts.drop(columns="HOLIDAY")

prediction = model.predict(X\_ts)

print(prediction)

# %%

plt.plot(Y\_tr,label = "Train Set")

plt.plot(Y\_ts,label = "Test Set")

plt.plot(prediction, label = "Forecast")

plt.legend()

plt.xlabel("Month")

plt.ylabel("Usage")

plt.title("Train, Test and the forecast after backward feature selection")

plt.xticks(rotation = 90)

plt.show()

# %%

pred\_error = Y\_tr.subtract(model.fittedvalues,axis=0)

print(pred\_error)

pred\_error = np.array(pred\_error)

# %%

rk = tx.stem\_plot(pred\_error,lag = 50,name = "Prediction Error")

print("ACF of resioduals:",rk)

# %%

# var\_pred = np.sqrt((1/(len(pred\_error)-len(X\_test.columns)))\*((sum(i\*i for i in pred\_error))))

print("MSE for prediction errors:",np.mean(np.square(pred\_error)))

print("The Variance for prediction error:",np.var(pred\_error))

for\_error = Y\_ts.USAGE- prediction

print("The MSE for forecast errors:",np.mean(np.square(for\_error)))

print("The variance for the forecast error:",np.var(for\_error))

Q = len(X\_train) \* np.sum(np.square(rk[20:]))

sm.stats.acorr\_ljungbox(for\_error, lags=[20], return\_df=True)

print("Q Value for MLR:",Q)

# %%

tx.arma\_dat(hourly\_usage\_df.USAGE,na =7,nb = 7)

# %%

model = sm.tsa.ARIMA(hourly\_usage\_df.USAGE, order=(2, 0, 0), trend="n", ).fit()

print(model.summary())

# Prediction

model\_hat = model.predict(start=0, end=len(X\_train) - 1)

pred\_error = X\_train.USAGE[1:] - model\_hat[:-1]

re = tx.stem\_plot(pred\_error, 20, name="Errors")

Q = len(X\_train) \* np.sum(np.square(re[20:]))

DOF = 20 - 2 - 0

alfa = 0.01

chi\_critical = chi2.ppf(1 - alfa, DOF)

if Q < chi\_critical:

print("The residual is white")

else:

print("The residual is NOT white")

lbvalue, pvalue = sm.stats.acorr\_ljungbox(pred\_error, lags=[20], return\_df=True)

print(lbvalue)

print(pvalue)

plt.figure()

plt.plot(X\_train.USAGE, "r", label="Train Data")

plt.plot(model\_hat, "b", label="Prediction")

plt.xlabel("Samples")

plt.ylabel("Magnitude")

plt.legend()

plt.xticks(rotation = 90)

plt.title("statsmodel ARMA process generation, parameter estimation and prediction")

plt.show()

#%%

forecast = model.predict(start=len(X\_train), end=len(hourly\_usage\_df))

for\_error = X\_test.USAGE-forecast

re = tx.stem\_plot(for\_error, 20, name="Errors")

Q = len(X\_test) \* np.sum(np.square(re[20:]))

DOF = 20 - 2 - 0

alfa = 0.01

chi\_critical = chi2.ppf(1 - alfa, DOF)

if Q < chi\_critical:

print("The residual is white")

else:

print("The residual is NOT white")

lbvalue, pvalue = sm.stats.acorr\_ljungbox(pred\_error, lags=[20], return\_df=True)

print(lbvalue)

print(pvalue)

plt.figure()

plt.plot(X\_test.USAGE, "r", label="Train Data")

plt.plot(forecast, "b", label="Forecast")

plt.xlabel("Samples")

plt.ylabel("Magnitude")

plt.legend()

plt.title("statsmodel ARMA process generation, parameter estimation and prediction")

plt.show()

#%%

print("MSE for prediction errors:",np.mean(np.square(pred\_error)))

print("The Variance for prediction error:",np.var(pred\_error))

print("MSE for forecast errors:",np.mean(np.square(for\_error)))

print("The Variance for forecast error:",np.var(for\_error))

#%%

diff1 = tx.difference(hourly\_usage\_df.USAGE,12)

tx.ACF\_PACF\_Plot(diff1,50)

# %%

# ARIMA(0,12,1)

model = sm.tsa.ARIMA(X\_train.USAGE, order=(0, 12, 1) ).fit()

print(model.summary())

# Prediction

model\_hat = model.predict(start=0, end=len(X\_train))

e = X\_train.USAGE - model\_hat

print("MSE for Prediction:",np.mean(np.square(e)))

print("Variance for Prediction:",np.var(e))

# forecast

forecast = model.predict(start = len(X\_train),end = len(hourly\_usage\_df))

e = X\_test.USAGE - forecast

print("MSE for forecast:",np.mean(np.square(e)))

print("Variance for Forecast:",np.var(e))

re = tx.stem\_plot(e, 20, name="Errors")

Q = len(X\_train.USAGE) \* np.sum(np.square(re[20:]))

DOF = 20 - 0 - 1

alfa = 0.01

chi\_critical = chi2.ppf(1 - alfa, DOF)

if Q < chi\_critical:

print("The residual is white")

else:

print("The residual is NOT white")

lbvalue, pvalue = sm.stats.acorr\_ljungbox(e, lags=[20], return\_df=True)

print(lbvalue)

print(pvalue)

plt.figure()

plt.plot(X\_train.USAGE, "r", label="True Data")

plt.plot(model\_hat, "b", label="Fitted Data")

plt.xlabel("Samples")

plt.ylabel("Electricity Usage in kWh")

plt.legend()

plt.xticks(rotation = 90)

plt.title("statsmodel ARIMA process")

plt.show()

#%%

plt.plot(X\_test.USAGE,label = "Test Data")

plt.plot(forecast,label = "Forecast")

plt.legend()

plt.xlabel("Months")

plt.ylabel("Electricity Usage in kWh")

plt.title("statsmodel ARIMA process")

plt.xticks(rotation = 90)

plt.show()

#%%

# SARIMA(0,1,1)

model=sm.tsa.statespace.SARIMAX(X\_train.USAGE,order=(0, 0, 0),seasonal\_order=(0,1,1,12))

results=model.fit()

print(results.summary())

prediction = results.predict(start = 0,end =len(X\_train.USAGE),dynamic = False)

plt.plot(X\_train.USAGE,label = "Train Data")

plt.plot(prediction,label ="Prediction")

plt.legend()

plt.xlabel("Month")

plt.ylabel("Electricity Usage in kWh")

plt.title("SARIMA train vs prediction")

plt.xticks(rotation = 90)

#%%

prediction\_error = X\_train.USAGE-prediction

print("MSE for SARIMA model (prediction):",np.mean(np.square(prediction\_error)))

print("Variance for SARIMA model (prediction):",np.var(prediction\_error))

# %%

forecast = results.predict(start = len(X\_train),end =len(hourly\_usage\_df.USAGE),dynamic = False)

plt.plot(X\_test.USAGE,label = "Test Data")

plt.plot(forecast,label ="Prediction")

plt.legend()

plt.xlabel("Month")

plt.ylabel("Electricity Usage in kWh")

plt.title("SARIMA train vs prediction")

plt.xticks(rotation = 90)

# %%

forecast\_error = X\_test.USAGE-forecast

print("MSE for SARIMA model (forecast):",np.mean(np.square(forecast\_error)))

print("Variance for SARIMA model (forecast):",np.var(forecast\_error))

# %%

re = tx.stem\_plot(forecast\_error, 20, name="Errors")

Q = len(X\_train.USAGE) \* np.sum(np.square(re[20:]))

DOF = 20 - 0 - 1

alfa = 0.01

chi\_critical = chi2.ppf(1 - alfa, DOF)

if Q < chi\_critical:

print("The residual is white")

else:

print("The residual is NOT white")

lbvalue, pvalue = sm.stats.acorr\_ljungbox(e, lags=[20], return\_df=True)

print(lbvalue)

print(pvalue)

# %%